Modeling of Dynamic Systems with Petri Nets and Fuzzy Logic

Lukas Windhager

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Lukas Windhager
aus Bad Ischl, Österreich

München, den 25.04.2013

Erstgutachter: Prof. Dr. Ralf Zimmer
Zweitgutachter: Prof. Dr. Fabian Theis
Tag der mündlichen Prüfung: 19.04.2013

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig und ohne unerlaubte Beihilfe angefertigt wurde.

Windhager, Lukas

Name, Vorname

Ort, Datum

Unterschrift Doktorand/in

Table of Contents

List of Figures

List of Tables

Zusammenfassung / Abstract

Zusammenfassung

Aktuelle Methoden zur dynamischen Modellierung von biologischen Systemen sind für Benutzer ohne mathematische Ausbildung oft wenig verständlich. Des Weiteren fehlen sehr oft genaue Daten und detailliertes Wissen über Konzentrationen, Reaktionskinetiken oder regulatorische Effekte. Daher erfordert eine computergestützte Modellierung eines biologischen Systems, mit Unsicherheiten und grober Information umzugehen, die durch qualitatives Wissen und natürlichsprachliche Beschreibungen zur Verfügung gestellt wird.

Der Autor schlägt einen neuen Ansatz vor, mit dem solche Beschränkungen überwunden werden können. Dazu wird eine Petri-Netz-basierte graphische Darstellung von Systemen mit einer leistungsstarken und dennoch intuitiven Fuzzy-Logik-basierten Modellierung verknüpft. Der Petri Netz und Fuzzy Logik (PNFL) Ansatz erlaubt eine natürlichsprachlich-basierte Beschreibung von biologischen Entitäten sowie eine Wenn-Dann-Regel-basierte Definition von Reaktionen. Beides kann einfach und direkt aus qualitativem Wissen abgeleitet werden. PNFL verbindet damit qualitatives Wissen und quantitative Modellierung.

Abstract

Current approaches in dynamic modeling of biological systems often lack comprehensibility, especially for users without mathematical background. Additionally, exact data or detailed knowledge about concentrations, reaction kinetics or regulatory effects is missing. Thus, computational modeling of a biological system requires dealing with uncertainty and rough information provided by qualitative knowledge and linguistic descriptions.

The author proposes a new approach to overcome such limitations by combining the graphical representation provided by Petri nets with the modeling of dynamics by powerful yet intuitive fuzzy logic based systems. The Petri net and fuzzy logic (PNFL) approach allows natural language based descriptions of biological entities as well as if-then rule based definitions of reactions, both of which can be easily and directly derived from qualitative knowledge. PNFL bridges the gap between qualitative knowledge and quantitative modeling.

Introduction

Chapter 1

Introduction

Systems biology as a field of research is concerned with studying of biological systems *as a whole* to understand their structure and dynamics ([1] p. 315, [2, 3, 4, 5, 6]). The parts of biological systems have been revealed to a great extent by various "-omics": genomics, proteomics, metabolomics, transcriptomics, etc ([7] with various links to data repositories). High-throughput technologies like micro-arrays [8, 9, 10], next-generation sequencing [11, 12, 13], and mass spectrometry [14, 15, 16, 17, 18] have been used to collect comprehensive data sets. Now, systems biology aims to infer the molecular networks and mechanisms that connect these parts and cause the phenotypic properties of biological systems [19].

One of the central questions of systems biology is how the complex dynamical behavior of a system emerges from individual interactions between biological molecules. Although this behavior emerges from their interplay, it is not predictable by studying the properties of single molecules or processes alone ([1] p. 310, [20, 21]). To study the dynamic behavior of a system, one has to investigate the dynamic interplay of molecules. This interplay can be studied *in silico* using computational models of the system.

1.1 Models of Biological Systems

Models of biological systems represent aspects of natural phenomena in a simplified way. The value of such models is determined by their usefulness to explain empirical observations in a way that makes predictions possible ([22] p. 90). Such predictions must allow verification or falsification by empirical observations ([22] p. 244). Every biological system - an organism, a cell, a cellular compartment, a signal transduction pathway, a part of a pathway - is constituted of entities and interactions. A model of such a biological system is an abstract, simplified representation of this system and includes abstract, simplified representations of entities and interactions.

Entities The abstract term entity may refer to any kind of species, actor, subject, or part of a biological system, as well as to all kinds of aggregations of these. Thus, the term entity might refer to a single, specific biological molecule, for example a specific transcription factor A_i which currently binds to a certain stretch of DNA in contrast to the other transcriptions factors of the sa-

me species $A_1, A_2, \ldots, A_n$ which are also denoted as individual entities. But the term entity might also refer to the whole set of transcription factors of the same kind A, which is an aggregation of the individual molecules. Biological molecules and sets of similar biological molecules can be seen as entities, but likewise can be parts of molecules like a gene as a part of the chromosomal molecule or aggregations of diverse molecules, like a cell as the aggregation of its various different constituting parts. Not only biological molecules or aggregates of these can be entities of a biological system, but also any other factors like for example the environmental conditions can be seen as an entity of a system as well.

At any given time, a biological system is in a certain state. The current state of a system is defined by the current states of the entirety of entities that are part of the system. In turn, the current state of an entity is defined by its current state with respect to properties of this entity. Each entity possesses a variety of properties that describe it. For example, a protein has a mass, a shape, a location within the cell. The set of proteins of a specific species has a concentration in mol or the proportion of phosphorylated proteins as a property. A gene has an expression level, which in fact describes the amount of the according mRNA, but nevertheless can be seen as a property of the gene. A cell has a type, e.g. blood or liver cell, and is in some phase of the cell cycle. The cellular environment has a temperature, a pH-level, etc. The current state of an entity with respect to a property can be specified by a numerical or linguistic value. Thus, the current state of an entity is defined by the collection of numerical/linguistic values of all its properties.

Interactions Entities in a biological system interact. Kinases phosphorylate other proteins or themselves, transcription factors regulate genes, mRNA is degraded, small molecules diffuse (both can be seen as a self-interactions or interactions with the environment), the environmental conditions influence enzymatic reactions, etc. In short, all kinds of (cellular) processes can be seen as interactions between pairs of entities or between sets of entities. Interactions between entities cause the dynamics of a biological system. Without them, the system would be static.

In general, a set of entities affects another set of entities via an interaction. The entities of the first set can be denoted effectors, those of the latter set targets. The same entity can be present in both sets. It can be effector and target at the same time. Through the interaction of effectors and targets, the state of the targets is changed depending on the current state of the effectors. This state change typically affects only a subset of the properties of the targets and is influenced only by the current states of a subset of properties of the effectors.

System Dynamics Entities and interactions primarily provide a static description of a system. They define relevant actors, their connectivity, possible paths of effect propagation, etc. If temporal and spatial dimensions are additionally considered, the dynamics of a system emerge from the interactions of entities. The current state of a system is defined by the states of its entities. Interactions like regulation of expression, enzymatic conversion, or transport processes cause state changes. A system's dynamics can be seen as trajectories of these state changes in time and space.

These dynamics are complex and can not be directly derived from a static description. Especially feedbacks cause a high degree of complexity. They can provide robustness and stability,

cause oscillations, or enhance, damp, or shutdown signal flow. Such a complex behavior can only be uncovered by studying the dynamics of a system. Thus, such a study is essential to gain insights into a system's properties and to allow predictions about its reaction to perturbations.

1.1.1 Abstraction and Representation

Every model of a biological system includes abstract representations of entities and interactions. Entities are represented by a selection of their properties and the associated values. Thus, a model includes a representation of the state of each entity with respect to some but typically not all properties. Most often, each entity is represented by a single state only, e.g. its concentration, expression level, or fold change. In computational models, states are typically represented as numerical values. The abstractions of interactions represent the effector-target relationships between entities. Most often, they also define quantitatively how the states of effector entities affect the states of target entities. In computational models, interactions are typically represented by mathematical functions.

Irrespective of the actual applied modeling technique (Section 1.2), all models can be graphically represented as networks of nodes and edges. As we consider biological systems as collections of entities and interactions, any biological system is inherently structured as a network, and likewise is implicitly or explicitly any model of such a system. How a network is actually derived from a model depends on the modeling technique and the desired type of network. For example, nodes could correspond to entities and edges to interactions, or both entities as well as interactions are nodes, e.g. as realized in Petri nets.

In general, only those states of entities and those interactions are included in a model that are of interest for the creator of the model. I.e. a model is built with a purpose and ideally includes exactly those entities and interactions that serve this purpose. Models are used for:

- Visualization of current knowledge in terms of entities and interactions that are relevant for a system.
- Structural analysis of the network, e.g. connectivity, degrees, clustering, or cycles.
- Analysis of a system's behavior given different initial states, parametrizations of functions, perturbations during execution, etc.
- Tests of hypotheses about the system by comparison to (new) experimental data

One can distinguish two basic types of models: static models and executable or dynamic models ([23] p. 10). A static model is equivalent to a graphical representation of entities and their interactions, for example as a network. It gives an idea about the connectivity of the underlying network and might as well provide basic information about the type of effector-target relationships, e.g. whether an effector acts activating or inhibiting on a target. They do not provide enough information to allow for a calculation of state changes, i.e. to predict future states based on the current states. Static models are suited to visualize or represent our knowledge of a system, for example as a figure in scientific literature. They can be very well suited to perform structural analysis of the model's underlying network.

Dynamic models are more complex models. They have to provide enough information to allow for the simulation of a system. States and interactions have to be defined such that given some initial states one has to be able to calculate state changes and thus future states. In dynamic models, interactions are represented as functions that can be evaluated based on the current states of effectors to calculate the state changes for their targets. These models can be used to learn how the observed behavior of a system arises, and can be used for *in silico* studies of the effects of perturbations. As dynamic models can be easily converted to static models without additional information, they can be used for anything that static models can be used for.

Models are built based on current knowledge and hypotheses about a system's entities and interactions. Hereby, the term knowledge denotes all kinds of available information, which can be roughly divided into two categories. First, experimental data, i.e. empirical observations of states and interactions during various experiments. Second, prior knowledge, i.e. knowledge about entities and interactions that can be obtained from scientific literature or other readily available sources. Examples for experimental data are concentration or expression levels of molecules at certain time points during experiments, binding affinities of molecules, conversion rates, etc. Prior knowledge might include reaction rates, the role of entities for example as transcription factors, knowledge about the chemical composition of a medium, etc. Models are built manually by an expert or user, by applying an automated or supervised reverse-engineering procedure, or by a combination of these. Model creation is usually done within a mine-model-mine cycle ([20],[23] pp. 17,[24, 25]):

- knowledge about a system is collected and hypotheses are formulated
- a model is created based on the knowledge and hypotheses
- new knowledge is collected and the model is compared to this knowledge
- hypotheses are modified and a new model is created, etc.

A good or adequate model of a biological system matches currently available prior knowledge and reproduces currently available experimental observations. If a model contradicts prior knowledge or data, this contradiction has to be justified. For example, if the network structure of the model contradicts prior knowledge but the simulated data matches experimental data, then one can conclude that the current knowledge about network structure is inaccurate and should be modified. Thus, the observed contradiction is justified by the explanation of experimental data and results in the modification of the current hypothesis about network structure.

1.1.2 Reverse-Engineering

One of the main goals of research of biological systems is to reverse-engineer networks from experimental data and to use them to investigate physiological and pathological mechanisms ([26] p. 222 , [27, 28]). Reverse-engineering is an inverse problem - it is the process of obtaining effector-target relationships between entities based on experimental observations. Although the actual molecular mechanisms of effector-target relations are hardly measurable, their effects can be observed in large-scale biochemical data. The observed effects allow conclusions about the underlying relations between entities [29].

Reverse-engineering of a dynamic model corresponds to creating a candidate model and using it to simulate data. If the simulated and experimental data are similar, the created dynamic model can be considered as an adequate representation of the true biological system, the reference system. Such reverse-engineered models are only approximations of the reference system, as the created models will be simplifications of the reference: relevant entities might be missing and the applied functions might be insufficiently complex. As measurements are typically incomplete, erroneous, and noisy, even a perfect model of the system might not be able to reproduce the experimental data perfectly.

The number of data-points necessary for reverse-engineering depends on the complexity of the (computational) modeling technique that is used for inference, the number of effectors that act on each target, the measurement error and signal-to-noise ratio, etc [27]. Very simple models that only represent functional associations between entities without defining any dynamics, i.e. static models, require the least amount of data. For example, pairwise correlation based clustering approaches require about $log(n)$ measurements of all n entities [30]. More involved approaches that allow a dynamic simulation of processes but still are significant simplifications of a system already require a number of measurements that grows exponentially with the number of involved entities. For example, 2^n measurements of n genes are necessary to obtain all functions of Boolean models, or about 2^K if the number of effectors is restricted to K [30]. Models that include continuous representations of states and allow for non-linear effects require even higher amounts of data than do linear or Boolean models, and typically the number of required measurements surpasses the number of those available [28, 29]. Functional relationships between effectors and targets can only be successfully assessed if the effector-induced changes in the target's states surpass the experimental and measurement noise (signal-to-noise ratio) [27]. All common high-throughput technologies suffer from noise: micro-array measurements are influenced by hybridization effects or dye-related signal correlation bias, next-generation-sequencing is influenced by unspecific sequence alignments and PCR bias, mass spectrometry is impaired by a lack of detection sensitivity and data reproducibility [31, 32, 33, 34, 35].

1.2 Computational Modeling in Bioinformatics

A wide variety of computational techniques has already been applied for modeling of biological systems. These techniques differ in their level of detail, their representation of entities and interactions, their graphical representation of the system, and allow for different types of analysis. The choice of a computational technique depends on the purpose of model building and the availability of biological knowledge [36, 37, 38].

Static models are used to represent the topology of a biological system, i.e. the network structure that emerges from entities and their interactions, and have for example been derived based on DNA-binding motifs [39, 40, 41], co-expression clustering [42, 43, 44, 45, 46], pairwise correlation, mutual information, or other correlation coefficients [47, 48, 49, 50, 51], nested effects [52, 53], or Bayesian inference [54, 55, 56].

Dynamic models are representations of a system that allow for the inference of dependent variables as functions of independent variables, where these variables are representations of states

of entities [27]. Thereby, these models can be used to predict the behavior of a system at future time points starting from an initial state that is derived from empirical data [27, 29]. In fact, the ability of a computational model to generate experimentally testable hypotheses is a major quality criterion. Dynamic modeling techniques have been extensively reviewed [57, 58, 59, 60, 61, 62, 63]. Some of the most common modeling techniques are based on Boolean or multi-valued logic [64, 65, 66, 67, 68, 69, 70], Petri nets [71, 72, 73, 74], or differential equations [75, 61, 76, 77, 78, 79]. Dynamic computational models have been widely used to study biological systems, for example apoptosis pathways [80, 81, 82], yeast cell cyclce [83, 84, 85, 86], mammalian cell cycle [87, 88, 89], mouse and drosophila embryonic pattern formation [90, 91, 92, 93], *C. elegans* and *A. thaliana* development [94, 95, 96, 97, 98], or biosynthesis of metabolites [99, 100, 101].

In the following, two techniques will be shortly discussed as they are relevant to this work: discrete logic models and models based on ordinary differential equations (ODE models).

1.2.1 Discrete Logic Models

Discrete logic models are radical simplifications of biological systems. In such models, it is assumed that entities can be described as being in two or more discrete states with respect to a property. If all entities are in one of two states, e.g. *on* or off, 1 or 0, *present* or *absent*, then discrete logic models are typically referred to as Boolean models, whereas if entities can be in one of three or more states, discrete logic models can be referred to as multi-valued logic models. The domain of discourse of any property is discretized using strict borders according to the number of desired states. For example, the domain of discourse of all concentrations, $\mathbb{R}_{\geq 0}$, can be discretized based on a threshold concentration c, such that all concentrations that are greater than c are described as *on* while all concentrations lower or equal than c are off. Hereby, *on* and off can be seen as sets, and all possible concentration values are assigned to either of them. If an entity is described as being in state *on* with respect to its concentration, then this means that the entity's concentration is greater than c and in the set *on*. A specific threshold, e.g. $c = 10$ mol, has to be chosen only if empirical data has to be discretized, e.g. to be compared to a model's predictions, or to be used to assign initial states. If a model is purely theoretical, such specific thresholds are only implicitly assumed. A discretization can be based on various aspects, but the most important are:

- Experimentally observed, typical abundances. If several typical value ranges for the abundance of an entity have been observed in experiments, then a discretization might reflect these different typical abundances. For example, an entity is either in state *on* (highly abundant) or off (lowly abundant), as applies for the independent variable y in **Figure 1.1**(left).

- Functional implications. If the concentration of an entity is in a certain value range, then it has a certain effect to its targets, and this effect is considerably different if the concentration is within another value range. For example, the state *on* of an activator implies that its target is in state *on* as well, while the state off implies that the target is off, as applies for the dependent variable x in **Figure 1.1**(left).

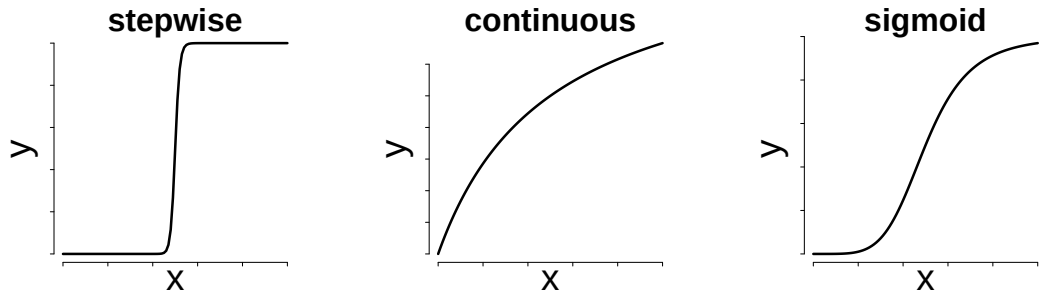


Figure 1.1 Discretization issues. (Left) A discretization of states, e.g. into *on* and off, can be meaningful if the values of the independent variable y (target's state) achieve clearly distinct values for most values of the independent variable x (effector's state) and if the intermediate values appear rarely. In such a case, it is obvious how a discretization should be performed. (Center) If there is a continuous dependency between y and x, a discretization is not adequate and results in spurious switches of discrete states although the underlying values changed only slightly. Furthermore, it is not clear how x or y should be discretized. (Right) There are not only extreme cases, but also mixed cases. Here, a discretization might be meaningful, but the aforementioned issue persists.

Thus, if an entity is described by discrete states, then a unique interpretation is inherently associated with each state, i.e. an interpretation with respect to e.g. abundance or functionality. The future states of entities are determined by the current states of their effectors through logical functions

$$y^{t+1} = f(x_1^t, \ldots, x_K^t)$$

Such logical functions correspond to rule tables that map all possible combinations of effector states to states of the target entity. For example, if in a Boolean model a target entity has K effectors, then each one the 2^K combinations of effector states is either mapped to *on* or off. Repeated evaluation of logical functions creates a trajectory of states. The dynamics of a system correspond to the trajectories of all entities.

A discretization of an intrinsically continuous property like a concentration is meaningful, if the concentration of an entity is actually found to be in few, distinct value ranges, and if the concentration switches between these value ranges so fast that intermediate concentrations values are rarely found or have no significant biological meaning. For example, a transcription factor could be found to be either expressed or not depending on the observed cell type [28]. This transcription factor could exhibit a switch-like, non-linear dependency on its effectors. Patterns of expressed and non-expressed transcription factors then specify cell types. These pattern may switch during differentiation, and such behavior can be successfully modeled by discrete logic models [30].

A major drawback of discrete logic models is that they can not represent intermediate states and thus they are not suited to model small or slow state changes and easily generate spurious results [28, 61]. For example, the concentration of a metabolite increases cumulatively as long as an according producing reaction occurs that converts an educt to the metabolite. If during this process, the metabolite's concentration exceeds the threshold c, the discretization switches immediately from off to *on* although the actual concentration changed only marginally (**Figure 1.1**).

1.2.2 Ordinary Differential Equations Models

Ordinary differential equations (ODE) models allow for a complex representation of systems. The states of entities are represented as continuous, real-valued variables, thus allowing for arbitrary small state changes. State changes during infinitesimal time intervals are defined by linear or non-linear differential equations.

$$y^{t+dt} = y^t + \frac{dy}{dt}, \qquad \frac{dy}{dt} = f(x_1^t, \ldots, x_K^t)$$

Thus, the new state of an entity is defined by summation of its current state and the state change. The effective state changes are determined by the current states of effectors, the applied reaction rate functions, and their parametrization. Due to the flexibility in the composition of reaction rate functions and their parametrization, complex dependencies of effector and target entities can be realized in ODE models.

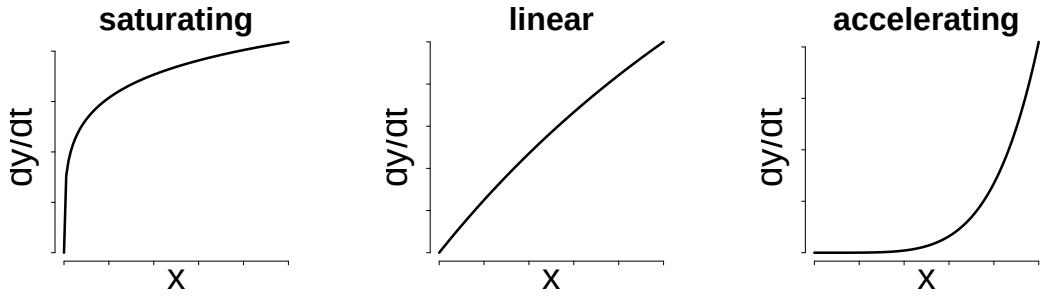


Figure 1.2 Obscured functionality. All three curves were created using the Hill equation with $x \in [0,1]$, $c = 1$, and $k = 3$. In all three cases, x has an increasing effect to dy/dt, but these effects are qualitative different. (Left) With parameter $n = 0.3$, the effect is saturating at high x. (Center) With $n = 1$, the effect is continuously increasing. (Right) With $n = 5$, the effect is accelerating with increasing x. This different qualitative behavior is not obvious from the equation alone, especially for users without an extended mathematical background.

A major drawback of ODE models is that the functional dependencies between entities are not easily interpretable, especially for users without an extended mathematical background [61]. The functionality of a system strongly depends on its parametrization. It may exhibit a significantly different behavior for different parametrizations, not only in a quantitative, but also in a qualitative way. For example, a commonly used and relatively simple representation of a functional dependency is based on the Hill equation

$$\frac{dy}{dt} = c \cdot \frac{x^n}{K^n + x^n}$$

Depending on the parameters K and n, the qualitative effect of x to dy/dt can be significantly different, ranging from saturating to accelerating (**Figure 1.2**) or from continuous to stepwise effects (like in **Figure 1.1**). Thus, not even a "simple" functional dependency between a single effector and its target can be comprehended by an examination of the according function alone,

much less can the full model. The interpretation of the model by a user is only facilitated if the model's creators explicitly provide additional information, or if the user performs additional analysis, for example by creating figures that display functional dependencies.

Analogously, ODE models do not offer an inherent interpretation of states. The continuous variables can achieve arbitrary values that do not stand out in any obvious aspects. An interpretation occurs not until the ODE model is additionally illustrated or is augmented by further analysis. For example, the different functional implications of the value ranges of independent variable x in **Figure 1.2** (right) only become apparent through investigating and explicitly visualizing its effect to dy/dt.

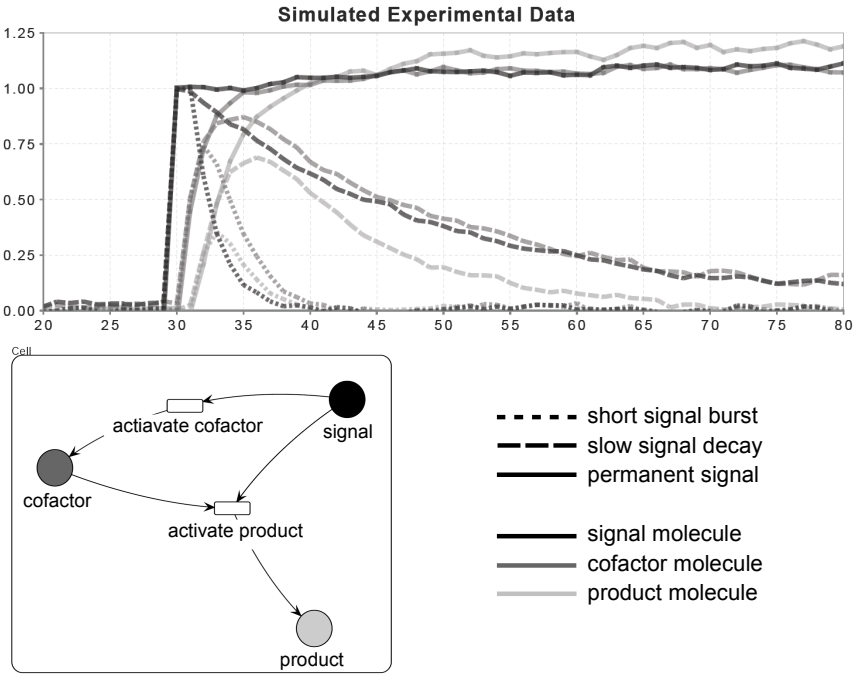
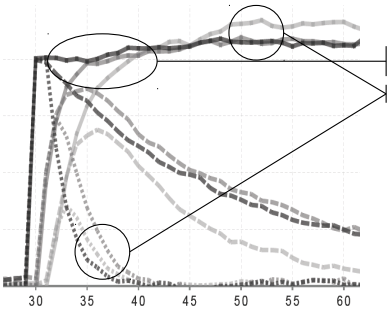


Figure 1.3 A simple feed-forward motif. A signal affects the concentration of a co-factor, while the concentration of a product is affected by both signal and co-factor (bottom left). The experimental data was simulated using a simple system of ODEs with mass-action kinetics, normal-distributed noise and three different decay rates for the signaling molecule (top). The reaction of the system after the addition of the signaling molecule at time point 30 is investigated. Dotted, dashed and solid lines indicate the measurements of the three different simulated experiments. Concentration (range axis) and time (domain axis) have arbitrary units.

1.3 Qualitative Knowledge

Knowledge about biological entities and of most kinds of biological data is typically incomplete and imprecise. This is caused by the size and complexity of biological systems and processes (biological noise) and by the inexactness of measurements, post-processing methods, or other kinds of technical noise [29]. So, when considering biological data, one typically works with average values characterized by smaller or larger variances. And most of the time, some kind of *best guess* (e.g. median) is considered as the truth, for example as the true concentration of a protein. Often enough, one not even knows the correct scales of biological data or has only a rough idea about the concentrations or other properties of biological entities. In addition to the uncertain data, scales and exact quantities may not even be of great importance in a biological system, for example with respect to its stability [59].

Let us consider a small exemplary system consisting of three proteins (**Figure 1.3**). The apparent information of the experiment consists of the measured concentrations during the observed time interval. However, the crucial point is to get an idea of the qualitative behavior of the system. The knowledge that can be deduced from this experiment is, that the concentration of a co-factor started to increase after a signal was inserted into the system. And only when both signal and co-factor are present, they induce the creation of the product. As soon as the signal is decayed, co-factor and product are decayed as well until they reach a low concentration. Such qualitative descriptions of a system can be converted into if-then rules that reflect the system's behavior, without the need to specify mathematical functions (**Figure 1.4**). Typically, such "natural language" based descriptions of knowledge explicitly or implicitly include or refer to some kind of discrete (and not continuous, real-valued) descriptions of objects, for example *low*, *medium*, and *high* concentrations.



"When the signal is inserted, first the cofactor is induced as then the product can be created.
Only if signal or cofactor are decayed, the product level decreases."

IF signal is *high* **THEN** cofactor is *high*.
IF signal is *low* **THEN** cofactor is *low*.

IF signal is *high* **AND** cofactor is *high* **THEN** product is *high*.
IF signal is *high* **AND** cofactor is *low* **THEN** product is *low*.
IF signal is *low* **AND** cofactor is *high* **THEN** product is *low*.
IF signal is *low* **AND** cofactor is *low* **THEN** product is *low*.

Figure 1.4 If-then rules describe processes. Linguistic or "natural language" descriptions of (observed) processes can easily be reformulated to if-then sentences. The behavior of the feed-forward motif observed in the three experiments can be described with sets of if-then rules defining the effects of the two processes indicated in **Figure 1.3**. Such if-then rule sets could be used to specify functions. They are easy to create and interpret.

We argue that it is not advisable to use ordinary sets to represent such linguistic terms, as the strict boundaries of ordinary sets are not intuitive. For example, the domain of possible concentrations of proteins can be normalized to the interval $[0, 1]$. The full interval could be divided

into three distinct sub-intervals $[0, 0.25)$, $[0.25, 0.75)$, and $[0.75, 1]$ such that concentrations within the borders of these sub-intervals are then characterized by the terms *low*, *medium* and *high* (**Figure 1.5**, left). Defining the set of highly concentrated proteins as "the set of proteins present at a level of more than 0.75" is unsatisfactory as this strict border might be artificial. It is difficult to argue, that a protein present at 0.750 is highly concentrated while it would *not* be highly concentrated at a level of 0.749. Therefore, it would be much more natural to define these sets with fuzzy borders (**Figure 1.5**, right).

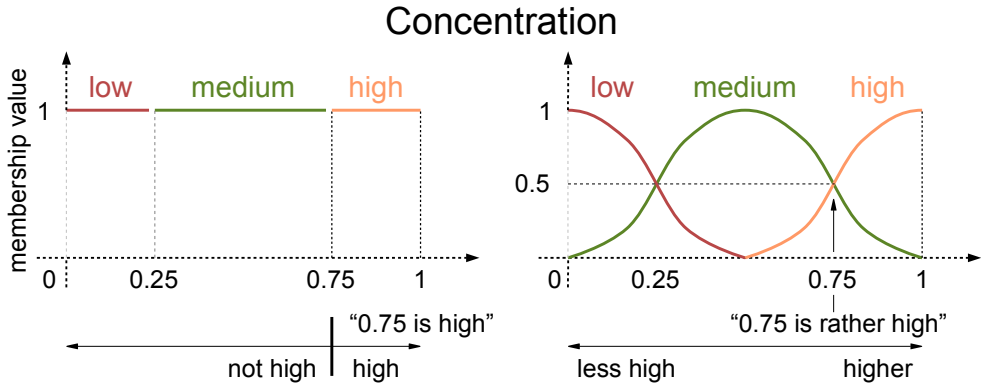


Figure 1.5 Discretization of concentrations. The (normalized) domain of possible protein concentrations can be divided in sharp intervals, i.e. discretized by ordinary sets *low*, *medium*, and *high* (left). A specific concentration, e.g. 0.75, is assigned to a single set. An inexact or "fuzzy" discretization can be done by allowing for a gradual change of memberships (right).

So, it seems quite promising to develop a computational approach which allows for a straightforward conversion of qualitative knowledge and descriptions into an executable model. Hereby, the first step is to find a suitable mathematical representation of discrete, inexact natural language terms that describe states of entities. The second step is to find a mathematical representation of functions that allows for a straightforward incorporation of qualitative knowledge based on natural language, and that will thereby facilitate an interpretation of the functionalities.

1.4 Our Contribution

In this work, we introduce a new modeling technique called PNFL modeling. It joins Petri nets and fuzzy logic (PNFL) in an innovative way and can benefit from properties of both technologies. Our intention is to develop a modeling technique that allows for the creation of inherently easy to interpret models, i.e. models that facilitate the understanding of states and processes. Still, these models should be powerful enough to simulate complex dynamic behaviors. PNFL models should be easier to interpret as ODE models and more powerful than discrete logic models.

This work is divided into two main parts that demonstrate that PNFL models are a suitable tool for modeling biological systems and that PNFL models can be used to reverse-engineer biological systems based on empirical data.

In part one, the Petri net and fuzzy logic modeling technique is introduced and defined and it is shown that PNFL is suited to model and analyze the behavior of biological systems. Hereby, the PNFL technique is stepwise introduced in the four main sections of part one. Section 2 describes how states of biological entities are represented using fuzzy sets (published in [102]). The use of fuzzy sets is illustrated by representing concentrations, fold-changes, and expression values. Section 3 describes how the functionalities of interactions are represented as fuzzy logic systems (published in [102]). It is shown how fuzzy logic systems can be designed such that they mimic commonly used functionalities like mass-action kinetics, Hill-functions, and logic gates. Section 4 defines PNFL models and shows how they can be used to model and analyze biological systems (published in [103]). Section 5 applies PNFL to model a prokaryotic transcription-translation system (published in [104]). A PNFL model is compared to an ODE model of the same system and it is shown how these models can be utilized to make predictions that allow for hypotheses testing.

In part two, we demonstrate how PNFL models can be reverse-engineered from empirical data. Therefore, we describe a non-deterministic genetic algorithm that can successfully reverse-engineer small PNFL models (Section 6, published in [105]). In addition, we present methods that improve prediction results for larger networks (Section 7) and post-process the predictions of non-deterministic reverse-engineering algorithms (Section 8).

To facilitate readability, we include basic definitions of fuzzy sets, fuzzy logic systems, etc within the sections of part one. An in-depth theoretical background about fuzzy logic and approximate reasoning is provided in the appendix (Section A).

Part I

Modeling with Petri Nets and Fuzzy Logic

Chapter 2

Fuzzy Sets Describe States of Biological Entities

At every point in time, biological entities like proteins, RNAs, cells, etc are in specific states with respect to certain properties. For example, such properties could be the concentrations of proteins in cells or compartments, fold-changes of mRNA species between different conditions, relative abundances of transcript isoforms, or the current phase of the cell cycle. To some extent, these properties can be measured or assessed in experiments. This gives us quantitative or qualitative information about the the current state of an entity. Of course, this information is often inaccurate or erroneous. Examples for obtained information:

1. Fluorescence measurements of proteins provide intensity levels, which might be converted into estimates of molar concentrations. This gives us absolute quantitative information about the proteins' concentrations at a certain time point, i.e. their *concentration states*.
2. RNA-seq or micro-array measurements of a transcriptome provide mRNA expression or intensity levels, which might be converted into fold-changes with respect to expression/intensity levels obtained from a reference experiment. This gives us relative quantitative information about the change of mRNA expression, i.e. the *fold-change states* of mRNAs.
3. Visual assessment of cells provides information about their phenotypes, i.e. whether these cells exhibit a certain phenotype or not. This gives us qualitative information about these cells, i.e. their *phenotypic state*.

All observations and the derived states have a domain of discourse D, i.e. their values are taken from a defined range or set of possible values. We denote values $x \in D$ as *crisp values*, in contrast to fuzzy values as introduced below. The domain of discourse depends on the type of observed property, the type of measurements or assessments, and the type of post-processing of the observed data. For example:

1. Concentrations might be in $\mathbb{R}_{\geq 0}$ in combination with a unit like mol, or might be relative, unit-less abundances in $[0, 1]$.
2. Expression changes might be log-fold-changes in $\mathbb{R}$.

3. Cell phenotypes might be taken from a discrete, finite categories set, e.g. cell cycle phases $\{G_0, G_1, S, G_2, M\}$.

Computational models of biological systems have to represent the current states of biological entities. Based on these representations, a system's behavior can be predicted or investigated (Section 1.1).

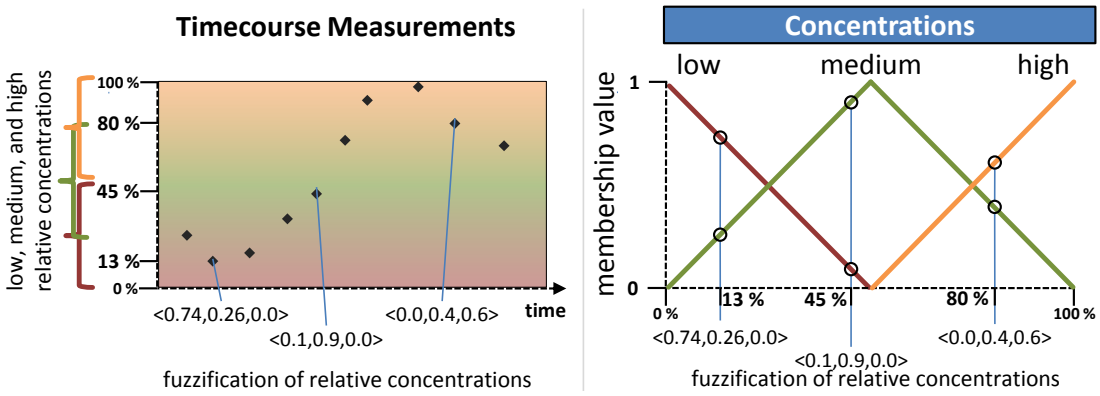


Figure 2.1 Fuzzy sets describe states. Assume that some measurements of protein concentration during an unspecified time interval are given (left panel). We quantify protein concentration as relative abundance. The domain of discourse comprises all real numbers in $[0,100]$ (unit %). The interval $[0,100]$ can be fuzzy discretized using three fuzzy sets that represent states of low (μ_{low}), medium (μ_{medium}), and high (μ_{high}) concentrations (right panel). The fuzzy sets map relative concentrations (x-axis) to membership values (y-axis). This three fuzzy sets constitute a fuzzy concept, and fuzzification of concentration $x \in [0,100]$ results in a fuzzy value $< \mu_{low}(x), \mu_{medium}(x), \mu_{high}(x) >$. This fuzzy value represents the current state of an entity with respect to a property (e.g. concentration) that in turn is described by the according fuzzy concept.

The representation of states with respect to certain properties can be based on *fuzzy sets* (Section A.2, [106]). They refer to aspects of the underlying biological property, e.g. "low concentrations", "medium concentrations", or "high concentrations", an can be interpreted as a fuzzy discretization of the according domain of discourse D. A fuzzy set μ maps all states $x \in D$ to the interval $[0,1]$, i.e. it assigns a *membership value* to each x that defines the degree of membership of x to the fuzzy set:

$$\mu : D \rightarrow [0,1]$$

The process of mapping a crisp value to a membership value is called *fuzzification* (Section A.2). If a state has a high degree of membership to a fuzzy set (high membership value) then the according aspect of the biological property applies to this state to a high degree. For example, fuzzy sets describing low concentrations assign high membership values to small $x \in D$ and small membership values to large $x \in D$ (**Figure 2.1**). Thus, a single fuzzy set provides a membership value that represents a state with respect to single aspect of a biological property.

To describe a state with respect to several properties, several fuzzy sets that have the same domain of discourse can be combined. Such combinations - called *fuzzy concepts* - can be represented as tuples $< \mu_1, \ldots, \mu_n >$. Such a fuzzy concept is a collection of fuzzy sets that describe the state of a biological entity with respect to the same property, e.g. concentration or fold-change. Fuzzyfication of a crisp value $x \in D$ by the fuzzy sets of a fuzzy concept gives us a vector $< \mu_1(x), \ldots, \mu_n(x) >$ with entries $\mu_i(x) \in [0,1]$. This vector is the actual representation of the state with respect to the fuzzy concept. This vector is called *fuzzy value*. Fuzzy values are used as inputs for fuzzy logic systems (Section 3). State changes and thus a systems behavior is computed based on the fuzzy value representation. The unified description of continuous as well as discrete properties, like concentrations and phenotypic states, by fuzzy values allows for a combination of such properties in fuzzy logic systems.

The number and shape of fuzzy sets that are joined to a fuzzy concept can be freely chosen according to design needs, e.g. depending on the type of biological property, available experimental data, or desired level of abstraction (Section 2.2). No matter how a fuzzy concept is designed, a fuzzy value $< \mu_1(x), \ldots, \mu_n(x) >$ derived by fuzzification of a crisp value $x \in D$ has the following properties:

1. $\forall i \in \{1, \ldots, n\} : \mu_i(x) \in [0,1]$
2. $\sum_{i=1}^{n} \mu_i(x) \in [0,n]$

Where property 1 corresponds to the definition of fuzzy sets and property 2 is derived from property 1 and the definition of fuzzy values. Fuzzy sets that constitute a fuzzy concept may overlap arbitrarily and a crisp value could have a high degree of membership to several fuzzy sets. However, we advise that fuzzy concepts should be designed such that $\sum_{i=1}^{n} \mu_i(x) = 1$ holds. On the one hand side, this guarantees that the domain of discourse is covered by fuzzy sets which is important for the use of the fuzzy concept in fuzzy logic systems (Section 3). On the other hand, this follows a quite natural and intuitive interpretation of a fuzzy value. This interpretation is that a biological entity has to be in *some* state, thus the fuzzy value has to be non-zero, and that the "membership potential" of a state is shared out to one or several fuzzy sets, thus the fuzzy value should sum to one.

A straightforward way to cover the full domain of discourse with fuzzy sets is to use unbounded fuzzy sets, i.e. fuzzy sets that assign a membership value of 1 to all crisp values larger (or smaller) than a given threshold (Section 2.1.4). Using triangle- or trapezoid-like fuzzy sets as defined in Section 2.1 to constitute a fuzzy concept ensures that the induced fuzzy values always sum to exactly one.

Each fuzzy value $< \mu_1(x), \ldots, \mu_n(x) >$ can be mapped to a crisp value $\bar{y} \in D$. This process is denoted *defuzzification*. In general, defuzzification derives a "most typical" crisp representative of a given fuzzy value. A common, intuitive, and computational simple defuzzification approach is height defuzzification (Section A.3.3). Here, the crisp value $\bar{y}$ is obtained by calculating a weighted average of typical representatives of those fuzzy sets that constitute the respective fuzzy concept:

$$\bar{y} = \frac{\sum_{j=1}^{n} \bar{y}_j \cdot \mu_j(x)}{\sum_{j=1}^{n} \mu_j(x)}$$

where $\bar{y}_j$ is the center of gravity of fuzzy set μ_j and used as its typical representative. Depending on the design of the fuzzy concept, the original crisp value x and the defuzzified value $\bar{y}$ can be identical. This is discussed in the following section, where we introduce some common shapes of fuzzy sets and according fuzzy concepts. These "most typical" crisp representatives can be straightforwardly used for visualization and are used to store the current state of an entity in PNFL models (Section 4).

2.1 Common Shapes of Fuzzy Sets

Any function $\mu : D \to [0,1]$ is a fuzzy set over the domain of discourse D. But for the application in modeling of biological systems, some basic shapes of fuzzy sets are especially suited. We will discuss triangle-, trapezoid-, and Gaussian-like fuzzy sets with the real numbers $\mathbb{R}$ as domain of discourse (**Figure 2.2**). Most often, e.g. when considering concentrations, expression levels, or fold-changes, the domain of discourse is naturally $\mathbb{R}$ or a subset thereof. Other domains of discourse of states of biological entities can generally be mapped to the domain of real numbers $\mathbb{R}$, e.g. numerical values $1,2,3,\ldots$ can be used to represent purely qualitative states like cell phenotypes. Further, we will discuss unbounded fuzzy sets, i.e. fuzzy sets that assign a membership value of 1 to all $x \in D$ that are larger or smaller than defined thresholds. Such fuzzy sets are especially suited to ensure that the domain of discourse is fully covered by a fuzzy concept.

2.1.1 Triangle-like Fuzzy Sets

The membership function of a triangle-like fuzzy set is defined as (**Figure 2.2A**):

$$\mu^{tri}(x) = \begin{cases} 0 & \text{if } x \leq l \\ \frac{x-l}{mp-l} \cdot m & \text{if } l < x < mp \\ m & \text{if } x = mp \\ \frac{r-x}{r-mp} \cdot m & \text{if } mp < x < r \\ 0 & \text{if } r \leq x \end{cases}$$

Parameter $m \in [0,1]$ specifies the maximum of μ^{tri} and is typically 1. Parameters $l, r, mp \in \mathbb{R}$ (left border, right border, maximum point) specify the shape and location of the triangle-like fuzzy set with respect to the domain of discourse. Defuzzification is usually performed using the center of gravity of the triangle-like fuzzy set. This corresponds to mp if the triangle is isosceles, i.e. if $mp - l = r - mp$. We advise to use mp as pseudo center of gravity also for non-isosceles triangular shapes, as this facilitates interpretation of defuzzified values. It is obvious that $l \leq x,y \leq r \wedge |x - mp| \neq |y - mp| \Leftrightarrow \mu^{tri}(x) \neq \mu^{tri}(y)$ holds for triangle-like fuzzy sets. Thus, the membership value of a state x is sensitive to changes in x as long as the fuzzy set covers x, i.e. if x is between l and r. From a functional perspective, using triangle-like fuzzy sets results in systems that are sensitive to state changes, i.e. small variations in states of effectors can be propagated to their targets. Furthermore, using triangle-like fuzzy sets is advantageous if a one-to-one mapping of a crisp value to its fuzzy value is desired.

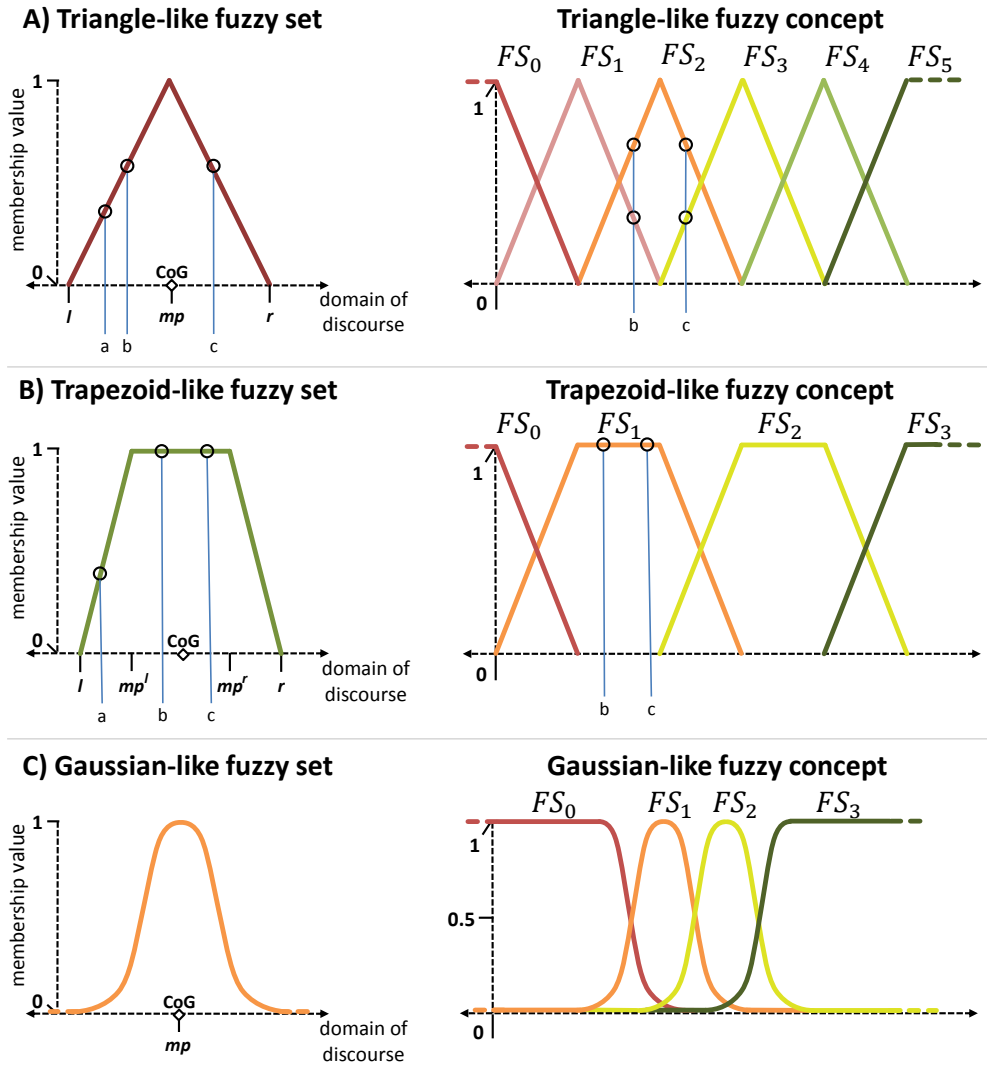


Figure 2.2 **Common shapes of fuzzy sets.** Bold lines show functions $\mu(x)$, except if $\mu(x) = 0$, then lines are omitted. Leftmost and rightmost fuzzy sets are designed as unbounded fuzzy sets (Section 2.1.4). **A)** Triangle-like fuzzy sets are defined by specifying their left and right border (l and r), as well as the point where $\mu^{tri}(x)$ reaches its maximum (mp). Small variations in crisp values cause changes in the corresponding membership values (e.g. crisp values a and b). Thus, triangle-like fuzzy set based representations are very sensitive to state changes. Although two different crisp values might have the same membership value with respect to a single fuzzy set, they will usually have different fuzzy values when a fuzzy concept of triangle-like fuzzy sets is considered (e.g. b and c). **B)** Trapezoid-like fuzzy sets are defined by specifying left and right border (l and r), as well as left and right maximum point (mp^l and mp^r). Although these fuzzy sets are sensitive to changes of crisp values if they are located between a border and the according maximum point (crisp value a), they are insensitive to changes between left and right maximum point (b and c). If a fuzzy concept is designed of trapezoid-like fuzzy sets as shown here, according fuzzy values are identical as well. **C)** Gaussian-like fuzzy sets are defined by their maximum point mp and a parameter w specifying the width of the bell-curve (not shown). They are a natural representation of noisy data centered around a specific value.

Proposition Assume a fuzzy concept consisting of n triangle-like fuzzy sets μ_i with increasing maximum points mp_i and left and right borders defined as follows: l_1 arbitrary, $r_1 = mp_2$. $\forall i$, $1 < i < n : l_i = mp_{i-1}$, $r_i = mp_{i+1}$. $l_n = mp_{n-1}$, r_n arbitrary (e.g. as in **Figure 2.2A**). Then fuzzification of any value $x \in [mp_1, mp_n]$ using a fuzzy concept as defined above, and subsequent height defuzzification with centers of gravity equal to the maximum points, results in a crisp value $\bar{y} = x$.

Proof Fuzzification of $x \in [mp_1, mp_n]$ results in a fuzzy value $< \mu_1(x), \ldots, \mu_n(x) >$. Per definition, at most two entries can be non-zero. Without loss of generality, assume that these are $\mu_i(x)$ and $\mu_{i+1}(x)$. Thus, the defuzzified value $\bar{y}$ is defined as:

$$\bar{y} = \frac{\sum_{j=1}^{n} \bar{y}_j \cdot \mu_j(x)}{\sum_{j=1}^{n} \mu_j(x)} = \frac{mp_i \cdot \mu_i(x) + mp_{i+1} \cdot \mu_{i+1}(x)}{\mu_i(x) + \mu_{i+1}(x)} \tag{2.1}$$

As per assumption $mp_i \leq x$ and $x \leq mp_{i+1}$, Equation 2.1 can be written as:

$$\bar{y} = \frac{mp_i \cdot \frac{r_i - x}{r_i - mp_i} + mp_{i+1} \cdot \frac{x - l_{i+1}}{mp_{i+1} - l_{i+1}}}{\frac{r_i - x}{r_i - mp_i} + \frac{x - l_{i+1}}{mp_{i+1} - l_{i+1}}} \tag{2.2}$$

Per definition of left and right borders, Equation 2.2 equals:

$$\bar{y} = \frac{mp_i \cdot \frac{mp_{i+1} - x}{mp_{i+1} - mp_i} + mp_{i+1} \cdot \frac{x - mp_i}{mp_{i+1} - mp_i}}{\frac{mp_{i+1} - x}{mp_{i+1} - mp_i} + \frac{x - mp_i}{mp_{i+1} - mp_i}} = mp_i \cdot \frac{mp_{i+1} - x}{mp_{i+1} - mp_i} + mp_{i+1} \cdot \frac{x - mp_i}{mp_{i+1} - mp_i} = x$$

Thus, a fuzzy concept as defined above allows a one-to-one mapping of a crisp value to its fuzzy value, **q.e.d.**

2.1.2 Trapezoid-like Fuzzy Sets

The membership function of a trapezoid-like fuzzy set is defined as (**Figure 2.2B**):

$$\mu^{tra}(x) = \begin{cases} 0 & \text{if } x \leq l \\ \frac{x - l}{mp^l - l} \cdot m & \text{if } l < x < mp^l \\ m & \text{if } mp^l \leq x \leq mp^r \\ \frac{r - x}{r - mp^r} \cdot m & \text{if } mp^r < x < r \\ 0 & \text{if } r \leq x \end{cases}$$

Parameter $m \in [0,1]$ specifies the maximum of μ^{tra} and is typically 1. Parameters $l, mp^l, mp^r, r \in \mathbb{R}$ (left border, left and right maximum point, right border) specify the shape and location of the trapezoid-like fuzzy set with respect to the domain of discourse. If the trapezoidal shape is symmetric, the center of gravity is at $mp^l + \frac{mp^r - mp^l}{2}$. As $mp^l < x, y < mp^r \wedge x \neq y \Leftrightarrow \mu^{tra}(x) = \mu^{tra}(y)$ holds, trapezoid-like fuzzy sets are robust against changes of x within the top base, i.e. different crisp values may have identical membership values. Trapezoid-like fuzzy sets can be used to define regions where the exact value of a state is not crucial or irrelevant for a systems behavior, e.g. when a model with a high level of abstraction is created.

2.1.3 Gaussian-like Fuzzy Sets

The membership function of a Gaussian-like fuzzy set is defined as (**Figure 2.2C**):

$$\mu^{gau}(x) = m \cdot e^{-\frac{(x-mp)^2}{2w^2}}$$

Parameter $m \in [0,1]$ specifies the maximum of μ^{gau} and is typically 1. Parameters $mp, w \in \mathbb{R}$ specify the maximum point and control the width of the curve and the center of gravity is at mp. Gaussian-like fuzzy sets are especially suited to represent noisy data centered around a specific value e.g. when representing fold-changes derived from expression data. Please note that $\forall x \in \mathbb{R} : \mu^{gau}(x) \neq 0$. Thus, membership values of Gaussian-like fuzzy sets are always non-zero.

2.1.4 Unbounded Fuzzy Sets

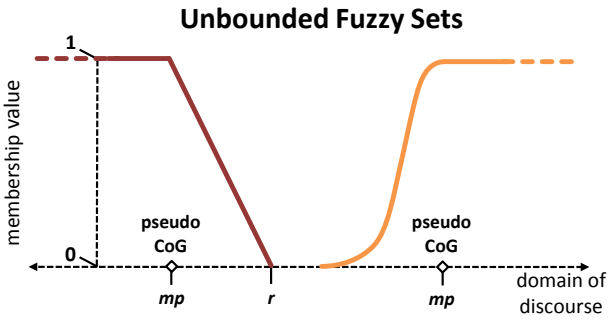


Figure 2.3 Unbounded fuzzy sets. If the domain of discourse is unbounded, e.g. the real numbers, and should be covered by fuzzy sets, it is reasonable to include unbounded fuzzy sets to a fuzzy concept (e.g. **Figure 2.5** left). These fuzzy sets assign a membership value of 1 to all crisp values above or below a defined threshold. As the centers of gravity for this type of fuzzy sets are in infinity, it is mandatory to define a pseudo center of gravity for defuzzification to allow meaningful results.

Unbounded fuzzy sets assign a membership value of 1 to all states of the domain of discourse that are larger (or smaller) than a given threshold. Given the real numbers as domain of discourse, these sets extend to either positive or negative infinity. We present two types of unbounded fuzzy sets: trapezoid-like unbounded fuzzy sets and Gaussian-like unbounded fuzzy sets (**Figure 2.3**). The membership functions of trapezoid-like unbounded fuzzy sets are defined as:

$$\mu^{tra}_{left}(x) = \begin{cases} 1 & \text{if } x \leq mp \\ \frac{r-x}{r-mp} & \text{if } mp < x < r \\ 0 & \text{if } r \leq x \end{cases} \qquad \mu^{tra}_{right}(x) = \begin{cases} 0 & \text{if } x \leq l \\ \frac{x-l}{mp-l} & \text{if } l < x < mp \\ 1 & \text{if } mp \leq x \end{cases}$$

Where μ^{tra}_{left} is unbounded left of mp and μ^{tra}_{right} is unbounded right of mp. The membership functions of Gaussian-like unbounded fuzzy sets are defined as:

$$\mu^{gau}_{left}(x) = \begin{cases} 1 & \text{if } x \leq mp \\ exp^{-\frac{(x-mp)^2}{2w^2}} & \text{if } mp < x \end{cases} \qquad \mu^{gau}_{right}(x) = \begin{cases} exp^{-\frac{(x-mp)^2}{2w^2}} & \text{if } x < mp \\ 1 & \text{if } mp \leq x \end{cases}$$

Note that the centers of gravity of unbounded fuzzy sets are either $-\infty$ or ∞, thus defuzzification can not be reasonably performed using these. We advise to use parameter mp as pseudo center of gravity instead.

2.2 The Design of Fuzzy Sets

The number, shapes, locations, and domain of fuzzy sets must be chosen according to design needs for a particular model or a particular set of observed data. They can be customized for each biological entity, e.g. concentrations of different proteins can be represented by different fuzzy concepts. Moreover, the same property of an entity can be represented by several different fuzzy concepts. For example, if it is an input to different fuzzy logic systems, i.e. when the biological entity is part of several different processes, and different representations are reasonable due to functional considerations. There are multiple aspects that influence the design of fuzzy sets:

Functional considerations Fuzzy concepts are used as inputs for fuzzy logic systems (Section 3). The number and shapes of fuzzy sets that represent a biological property determine the possible design and power of fuzzy logic systems. Thus, one of the most important design considerations is guided by the intended functionality. For example, it is often reasonable to design fuzzy sets such that they represent concentrations that imply a similar functional behavior (**Figure 2.4**).

Type of biological property. The fuzzy set design depends on the type of biological entity and the type of property that should be represented. For example, when creating a fuzzy representation of concentrations, one would introduce fuzzy sets describing different concentration levels, while when creating a fuzzy representation of fold-changes, one would introduce fuzzy sets describing down-regulation, wild-type, and up-regulation (**Figure 2.5** left).

Range of observed data. The range of actually observed data (in contrast to the domain of discourse) guides fuzzy set design (e.g. see fuzzy set design in Section 5). For example, if protein concentrations between 0 and 10 nM were observed but never higher concentrations, then one might use several fuzzy sets to fuzzy discretize the domain [0,10], e.g. to represent nearly absent protein, low, medium, and high protein concentration. But a single fuzzy set covering the domain [10,∞] might be sufficient to represent exceptionally high concentrations.

Desired level of abstraction. The more fuzzy sets are used, the more fine-grained is the fuzzy discretization. This might be useful for interpretation and to fine-tune the quantitative behavior of a system, but most often a qualitatively adequate functionality can be achieved with very few fuzzy sets (e.g. see fuzzy set design in Section 4.2).

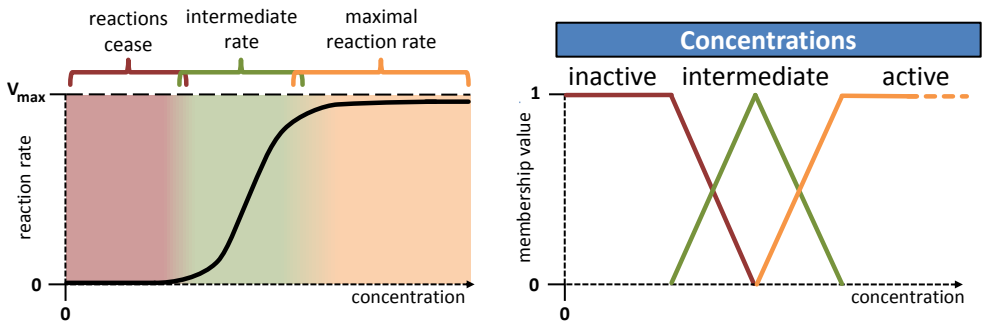


Figure 2.4 Fuzzy sets designed to represent functional properties. Cooperative binding of ligands to receptors might lead to sigmoid-shaped reaction rates depending on ligand concentration. If the concentration is too low (nearly) no processes occur. The reaction rate saturates at high ligand concentrations, i.e. a further increase of concentration does not increase reaction rates. This concentration dependent functional behavior can be used as a guideline for fuzzy set design. If the concentration is below or above an intermediate concentration range, the fuzzy values are close to $< 1,0,0 >$ (inactive) and $< 0,0,1 >$ (active). Further decrease or increase of concentration does not change the fuzzy value representation.

Detection limits, precision, and robustness. Measurement methods might not be able to reliably detect concentrations if they are too low (or too high), thus a fine-grained distinction of these concentration levels (e.g. by several triangle-like fuzzy sets) is not meaningful and a single trapezoid-like fuzzy set could be used instead to cover a broad range. Replicate measurements could differ due to technical noise or biological variances. If large variations are found, wide fuzzy sets could be used, while narrow ones can be used otherwise.

Typical values. Fuzzy sets can be designed to represent typical values of a property, for example expression levels of a certain mRNA species in two cell types. One cell type might have a low expression level of this mRNA while the other has a high expression level. These expression levels that are typical for a certain cell type could be modeled as Gaussian- or trapezoid-like fuzzy sets centered at the respective expression level (**Figure 2.5** right).

2.3 Discussion

A fuzzy set describes an aspect of a biological property or concept, e.g. low concentrations, medium concentrations, or high concentrations, and can be interpreted as part of a fuzzy discretization of the respective domain of discourse, e.g. a discretization of all possible concentrations. We denote a collection of fuzzy sets that describe the same property as fuzzy concept. The current state of an entity with respect to a property is represented by a vector of membership values, a fuzzy value. The membership values specify to which degree the different aspects currently apply to the state of the entity.

In the following Section 3, we will describe how fuzzy logic systems can be used to represent the interactions of a system. Fuzzy sets are used in these fuzzy logic systems to describe the

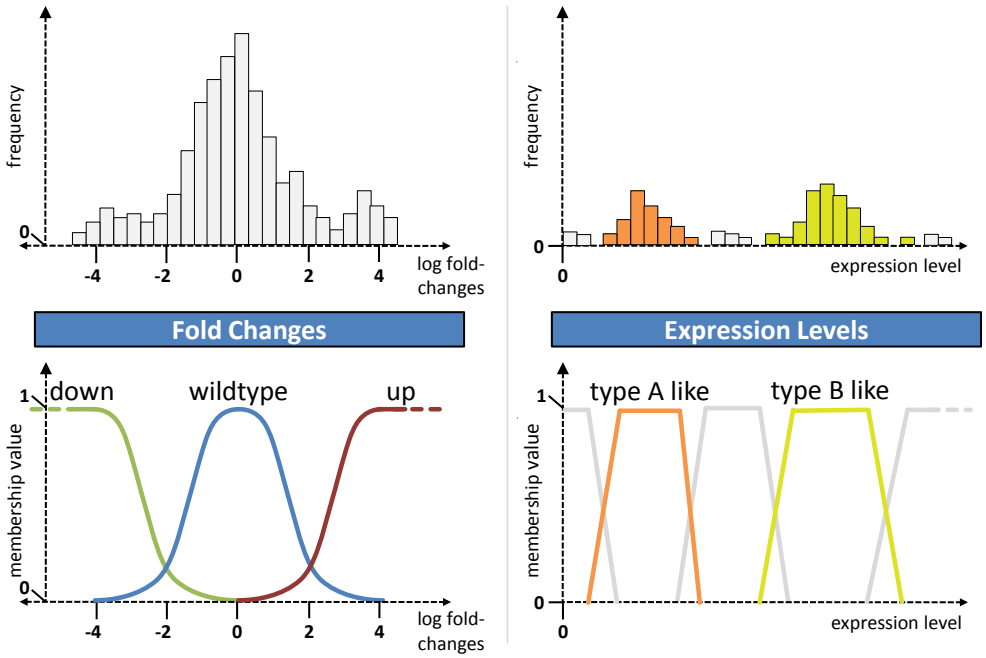


Figure 2.5 Designing fuzzy sets. Fuzzy sets are designed according to the type of observed data. If expression data is given as log-fold-changes, it might be reasonable to distinguish between wild-type expression states (assuming noise) and differentially regulated states (left panel). If some typical values are observed in experiments, according fuzzy sets could be defined. For example, if the expression level of a mRNA species is specific for certain cell types (right panel). Fuzzy concepts should cover the full domain of discourse. This can be achieved by defining appropriate unbounded fuzzy sets.

current states of the effectors of the interactions. Hereby, the same effector may be part of several interactions and may be described using different fuzzy sets in each of the according fuzzy logic systems. So the concentration of the same effector may be described using several fuzzy concepts at the same time, each of which are used in different fuzzy logic systems. For examples, see **Figure 4.9** in Section 4.2 and Section 5.2.2.

Furthermore, the description of a state has a direct influence to the functionality of the model. By changing the definitions of fuzzy sets, the outcomes of simulations can be changed without changing the rule bases of fuzzy logic systems (see Section 3). The unified description of continuous as well as discrete properties like concentrations and phenotypic states by fuzzy values allows for a straightforward combination of such properties in fuzzy logic systems.

The presented common shapes of fuzzy sets (Section 2.1) can be used to reflect the impact of state changes of crisp values to a model. If small changes of a state are meaningful and should affect a model, then triangle-like fuzzy sets are a suitable representation, as the derived fuzzy values reflect these small changes. In particular, we have shown that fuzzy concepts can be designed such that there is a one-to-one correspondence of fuzzy value and crisp value representation, and thus there is no information loss by fuzzification. If small state changes of an entity have no relevant effect to a system, then this can be reflected by using trapezoid-like fuzzy sets. Such fuzzy sets assign the same membership value to all crisp values within a certain range. Thus, they "filter" small state changes.

A comparison to ODE and discrete logic models A drawback of ODE models is that they do not offer a direct interpretation of states. ODE models represent states of entities by real-valued variables only (Section 1.2.2). For example, although we may know the numerical value of the concentration of a transcription factor, we do not know whether this concentration is relatively low or high or typical for a certain cell type, or whether at this concentration the transcription factor has a considerable effect on the transcription rates of its targets or not. This information is hidden within the ODE model and only becomes apparent if the model is simulated and the resulting data sets are interpreted. Or, such information has to be provided as additional information. In contrast, a representation based on fuzzy sets inherently includes an interpretation of the described states with respect to a biological relevant aspect, for example with respect to the functional impact, typical values, etc (Section 2.2). As the same entity, for example a transcription factor, can be represented by several fuzzy concepts at the same time, one can provide interpretation with respect to different aspects at the same time.

Discrete logic models represent states using two or more categories, e.g. *on* and *off*, or *present* and *absent*. These categories correspond to an interpretation of the state, but the strict borders of these categories are unnatural and unintuitive, as was discussed in Section 1.2.1. Furthermore, the state of an entity has to be exactly in one of the allowed states. In contrast, fuzzy sets can be designed with arbitrary fluent transitions of one category to the next. Thus, there are no abrupt changes of the interpretation of a state and an entity can be in a meaningful intermediate state. The main advantage of using fuzzy sets to describe states is the inherently provided interpretation of these states. This significantly facilitates understanding of models while not suffering the drawbacks of unnatural strict discretization borders.

Author's contribution Fuzzy set are an established concept [106]. The author describes various aspects of their application to represent biological data.

Chapter 3

Fuzzy Logic Systems Give Functionality to Interactions

Interactions between biological entities represent processes that influence the future state of the target entities based on the current state of the effector entities. Computational models mimic interactions by functions that operate on the computational representations of states (Section 1.1). These functions map the current states of effectors (inputs) to new states or state changes of targets (outputs). Repeated application of functions creates a trajectory of state changes which describes the dynamics of a system.

Fuzzy logic systems are functions that can be used to describe such state changes. They map input crisp values representing states to an output crisp value:

$$fls : D_1 \times \ldots \times D_n \rightarrow D_y$$

where D_i is the domain of discourse of input crisp value x_i and D_y is the domain of discourse of the output crisp value $\bar{y}$. This process of mapping can be split into the intermediate steps of fuzzification, fuzzy inference, and defuzzification (**Figure 3.1**). Fuzzification and defuzzification of input crisp values have been described in Section 2. For each input crisp value a fuzzy concept is used for fuzzification, denoted antecedent fuzzy concept. Fuzzy inference is specified using a set of if-then rules - the *rule base* - of the form:

$$R_j : \; IF \; x_1 \; is \; A_{j,1} \; AND \; x_2 \; is \; A_{j,2} \; AND \; \ldots \; AND \; x_{n_j} \; is \; A_{j,n} \; THEN \; y \; is \; B_j$$

Where $x_i \in D_i$ are the input crisp values, $A_{j,i}$ is a fuzzy set taken from fuzzy concept A_i used to fuzzify the i'th input crisp value in rule j (*antecedent fuzzy set*), and B_j specifies the *consequent fuzzy set*. Consequent fuzzy sets are taken from a consequent fuzzy concept B.
Each if-then rule derives a *conclusion fuzzy set* $\mu_{B_j^*} : D_y \rightarrow [0,1]$ based on input crisp values x_i and the according consequent fuzzy set μ_{B_j}. The inference of the conclusion fuzzy sets $\mu_{B_j^*}$ is based on approximate reasoning theory. We will briefly summarize the main ideas. For details, see Section A. In approximate reasoning the (classic) truth values 0 and 1 are extended to degrees of truth taken from the interval $[0,1]$. Fuzzy sets are used to quantify the degree of truth of elements of their domain of discourse. The degree of truth of the premise 'x_i is $A_{j,i}$' is identical to the

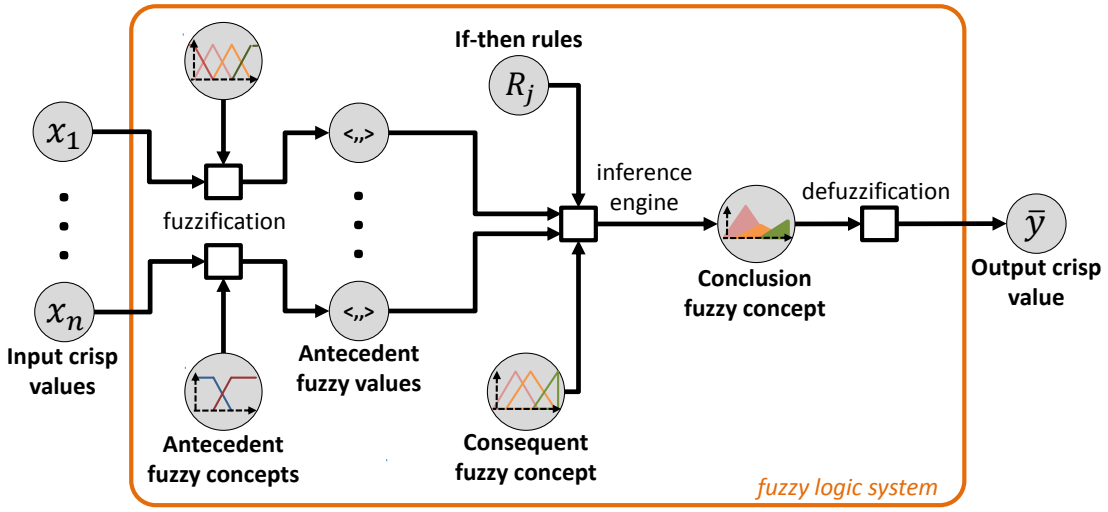


Figure 3.1 Fuzzy logic system flow-chart. A fuzzy logic systems maps input crisp values to an output crisp value. First, input crisp values are fuzzified based on antecedent fuzzy concepts. Antecedent fuzzy values are subsequently used by the inference engine to derive a conclusion fuzzy concept containing weighted versions of fuzzy sets taken from the consequent fuzzy concept. The inference engine is specified by a set of if-then rules. The conclusion fuzzy concept is defuzzified to obtain the output crisp value.

membership value $\mu_{A_{j,i}}(x_i)$. If-then rules define which conclusion might follow from the defined premises. Using classical set notion, a rule could be read as: 'If x is any element of A, then y is any element of B', or using fuzzy set notion: 'If x is A ($\hat{=}\mu_A(x)$) then y is B ($\hat{=}\mu_B(y)$)'. The degree of truth of the conclusion is inferred from the degrees of truth of the premises. Generally, it holds that the higher the degrees of truth of the premises, the higher is the degree of truth of the conclusion. The degree of truth μ_{B^*} is derived from the degrees of truth of the premises and the antecedent fuzzy set μ_B. In general, μ_{B^*} is a capped or scaled image of μ_B. Thus, $\forall y : \mu_{B^*}(y) \leq \mu_B(y)$, and $\mu_{B^*}(y) = \mu_B(y)$ only if the truth values of all premises are exactly 1. The conclusion fuzzy set defines a degree of truth to all elements of its domain of discourse. It quantifies the validity of each element as a conclusion. The formal notation of a rule base is:

$$\mu_{B^*}(y) = \bigcup_{j=1}^{m} \mu_{B_j^*}(y) = \bigcup_{j=1}^{m} \left(\bigcap_{i=1}^{n} \mu_{A_{j,i}}(x_i') \star \mu_{B_j}(y) \right) \tag{3.1}$$

where $\cup$ is a disjunction operation specified by a T-conorm, and $\cap$ and $\star$ are typically the same conjunction operation specified by a T-norm (Section A.2.1). According to Equation 3.1, each individual rule has to be composed of a conjunction of positive literals, each of which corresponds to a fuzzified premise.

Despite this strict formal requirement for the composition of rule bases, individual rules can be designed freely according to design needs as long as they can be converted to a disjunction of conjunctions of positive literals, each explicitly comprising all premises exactly once. This conversion can be performed by splitting of rules, special fuzzy set design, and intersection,

union, and complements of fuzzy sets as defined in Section A. In the following, we provide several examples of rule design and according conversions:

Negated premises A rule which contains a negative literal

$$R_1 : \mathit{IF\ NOT\ } x \mathit{\ is\ } A \mathit{\ THEN\ } y \mathit{\ is\ } B$$

can be written formally as

$$\neg\mu_A(x) \star \mu_B(y)$$

which can be converted to

$$(1 - \mu_A)(x) \star \mu_B(y)$$

which is a conjunction of positive literals. Instead of using fuzzy set μ_A as antecedent fuzzy set, the complement fuzzy set $(1 - \mu_A)$ is used.

Missing premises A rule base with a rule which misses a premise

$$R_1 : \mathit{IF\ } x_1 \mathit{\ is\ } A_1 \mathit{\ AND\ } x_2 \mathit{\ is\ } A_2 \mathit{\ THEN\ } y \mathit{\ is\ } B_1$$
$$R_2 : \mathit{IF\ } x_2 \mathit{\ is\ } A_2 \mathit{\ THEN\ } y \mathit{\ is\ } B_2$$

can be written formally as

$$\mu_{A_1}(x_1) \star \mu_{A_1}(x_2) \star \mu_{B_1}(y) \circ \mu_{A_2}(x_2) \star \mu_{B_2}(y)$$

which can be converted to

$$\mu_{A_1}(x_1) \star \mu_{A_2}(x_2) \star \mu_{B_1}(y) \circ \mu_1(x_1) \star \mu_{A_2}(x_2) \star \mu_{B_2}(y)$$

with $\mu_1 : \mathbb{R} \rightarrow 1$ is a fuzzy set that maps all elements of its domain of discourse to 1 and $\circ$ is a disjunction operator (T-conorm). Thus, rule R_2 is still independent of premise x_1, although now all rules comprise all premises exactly once.

Disjunctions A rule which contains a disjunction of literals

$$\mathit{IF\ } x_1 \mathit{\ is\ } A_1 \mathit{\ OR\ } x_2 \mathit{\ is\ } A_2 \mathit{\ THEN\ } y \mathit{\ is\ } B$$

can be written formally as

$$(\mu_{A_1}(x_1) \circ \mu_{A_2}(x_2)) \star \mu_B(y)$$

which can be split into two separate rules

$$\mu_{A_1}(x_1) \star \mu_1(x_2) \star \mu_B(y) \circ \mu_1(x_1) \star \mu_{A_2}(x_2) \star \mu_B(y)$$

which are a disjunction of conjunctions.

The disjunction of conclusion fuzzy sets of all if-then rules constitutes a conclusion fuzzy concept $\mu_{B^*}(y)$. The conclusion fuzzy concept is subsequently defuzzified which results in a single output crisp value (Section A.3.3). This output crisp value is interpreted as the *most typical* crisp representative of the conclusion fuzzy concept. When using height defuzzification, the output crisp value corresponds to the weighted average of the centers of gravity of conclusion fuzzy sets. Hereby, the membership values of the centers of gravity, given by the conclusion fuzzy sets, are used as weights:

$$\bar{y} = \frac{\sum_{j=1}^{m} \bar{y}_j \mu_{B_j^*}(\bar{y}_j)}{\sum_{j=1}^{m} \mu_{B_j^*}(\bar{y}_j)}$$

where $\bar{y}_j$ are the centers of gravity of conclusion fuzzy sets $\mu_{B_j^*}$. Note that the center of gravity of a conclusion fuzzy set equals the center of gravity of the according consequent fuzzy set. The values $\mu_{B_j^*}(\bar{y}_j)$ can be represented as a fuzzy value, denoted conclusion fuzzy value. Using product inference and height defuzzification (Section A.3.3), a fuzzy logic system can be written very compactly:

$$\bar{y} = fls(x_1, \ldots, x_n) = \frac{\sum_{j=1}^{m} \bar{y}_j \mu_{B_j^*}(\bar{y}_j)}{\sum_{j=1}^{m} \mu_{B_j^*}(\bar{y}_j)} = \frac{\sum_{j=1}^{m} \bar{y}_j \cdot \prod_{i=1}^{n} \mu_{A_{j,i}}(x_i)}{\sum_{j=1}^{m} \prod_{i=1}^{n} \mu_{A_{j,i}}(x_i)} \tag{3.2}$$

In fuzzy logic theory, several different types of inference and defuzzification are known and can be used to build fuzzy logic systems [107]. Nevertheless, here we suggest to use product inference and height defuzzification for modeling of biological systems. When using product inference, changes of fuzzified values in the antecedent of a rule always affect the conclusion fuzzy sets. I.e. if the fuzzified value of one premise is decreased, then the conclusion fuzzy set is decreased as well, and *vice versa*. In contrast, when using minimum inference, changes of fuzzified values have no effect to the conclusion fuzzy set, if the minimum of fuzzified values is not changed. Thus, using product inference follows the intuition that changes of the degrees of truth of premises affect the degree of truth of the conclusion. The main reason for height defuzzification is that it is a computational simple technique. The centers of gravity of conclusion and consequent fuzzy sets are identical, and as consequent fuzzy sets are known beforehand, centers of gravity do not need to be re-computed (Section A.3.3). Additionally, the only necessary information about consequent fuzzy sets is the location of their centers of gravity, i.e. the shapes of consequent fuzzy sets are irrelevant. Thus, when designing rules for fuzzy logic systems, simple singleton fuzzy sets can be used as consequent fuzzy sets:

$$\mu_{mp}^{s}(x) = \begin{cases} 1 & \text{if } x = mp \\ 0 & \text{else} \end{cases}$$

which have their center of gravity at mp. Throughout this document, product inference and height defuzzification are used for all fuzzy logic systems. Most often, rule bases will be designed such that they contain a rule for all combinations of antecedent fuzzy sets. E.g. if premise x_1 is fuzzified using fuzzy concept $< \mu_{A_1}, \mu_{A_2} >$ and premise x_2 is fuzzified using $< \mu_{A_3}, \mu_{A_4} >$, then

combining these antecedent fuzzy sets results in four different rules. A simple and clear way of representing such a rule base with up to two premises is using a tabular notation:

$$
\begin{aligned}
&R_1: \mathit{IF}\ x_1\ \mathit{is}\ A_1\ \mathit{AND}\ x_2\ \mathit{is}\ A_3\ \mathit{THEN}\ y\ \mathit{is}\ B_1\\
&R_2: \mathit{IF}\ x_1\ \mathit{is}\ A_2\ \mathit{AND}\ x_2\ \mathit{is}\ A_3\ \mathit{THEN}\ y\ \mathit{is}\ B_2\\
&R_3: \mathit{IF}\ x_1\ \mathit{is}\ A_1\ \mathit{AND}\ x_2\ \mathit{is}\ A_4\ \mathit{THEN}\ y\ \mathit{is}\ B_3\\
&R_4: \mathit{IF}\ x_1\ \mathit{is}\ A_2\ \mathit{AND}\ x_2\ \mathit{is}\ A_4\ \mathit{THEN}\ y\ \mathit{is}\ B_4
\end{aligned}
\quad \Leftrightarrow \quad
\begin{array}{|cc|cc|}
\hline
 & & \multicolumn{2}{c|}{x_1} \\
 & & A_1 & A_2 \\
\hline
x_2 & A_3 & B_1 & B_2 \\
 & A_4 & B_3 & B_4 \\
\hline
\end{array}
$$

3.1 A Numerical Example

In the following, an example for a fuzzy logic system is given. Consider a fuzzy logic system $fls : [0,1] \times [0,1] \rightarrow [0,1]$ that describes a reaction rate as a function of two educt molecules. This fuzzy logic system maps two input crisp values x_1 and x_2 from the domain of discourse $[0,1]$ to an output crisp value $\bar{y} \in [0,1]$. Let the values of the crisp inputs be:

$$x_1 = 0.15 \quad \text{and} \quad x_2 = 0.7$$

These input crisp values are described using two different antecedent fuzzy concepts as defined in **Figure 3.2A**. Input x_1 is described as being either *absent* or *present*, while input x_2 is described as being present at either *low*, *medium*, or *high* concentration. Fuzzification results in the antecedent fuzzy values:

$$
\begin{aligned}
< \mu_{absent}(x_1), \mu_{present}(x_1) > &= < 0.8, 0.2 >\\
< \mu_{low}(x_2), \mu_{medium}(x_2), \mu_{high}(x_2) > &= < 0.0, 0.6, 0.4 >
\end{aligned}
$$

As we use height defuzzification, we only need to specify the centers of gravity of the consequent fuzzy sets. Thus, we define consequent fuzzy sets *ceased*, *slow*, and *fast* as singleton fuzzy sets with centers of gravity $\bar{y}_{ceased} = 0$, $\bar{y}_{slow} = 0.5$, and $\bar{y}_{fast} = 1$ (**Figure 3.2B**). The rule base of fls is defined by a set of if-then rules:

$$
\begin{aligned}
&R_1 : \mathit{IF}\ x_1\ \mathit{is\ absent}\ \ \mathit{AND}\ x_2\ \mathit{is\ low}\ \ \ \mathit{THEN}\ y\ \mathit{is\ ceased}.\\
&R_2 : \mathit{IF}\ x_1\ \mathit{is\ absent}\ \ \mathit{AND}\ x_2\ \mathit{is\ medium}\ \mathit{THEN}\ y\ \mathit{is\ ceased}.\\
&R_3 : \mathit{IF}\ x_1\ \mathit{is\ absent}\ \ \mathit{AND}\ x_2\ \mathit{is\ high}\ \ \ \mathit{THEN}\ y\ \mathit{is\ ceased}.\\
&R_4 : \mathit{IF}\ x_1\ \mathit{is\ present}\ \mathit{AND}\ x_2\ \mathit{is\ low}\ \ \ \mathit{THEN}\ y\ \mathit{is\ ceased}.\\
&R_5 : \mathit{IF}\ x_1\ \mathit{is\ present}\ \mathit{AND}\ x_2\ \mathit{is\ medium}\ \mathit{THEN}\ y\ \mathit{is\ slow}.\\
&R_6 : \mathit{IF}\ x_1\ \mathit{is\ present}\ \mathit{AND}\ x_2\ \mathit{is\ high}\ \ \ \mathit{THEN}\ y\ \mathit{is}\ fast.
\end{aligned}
$$

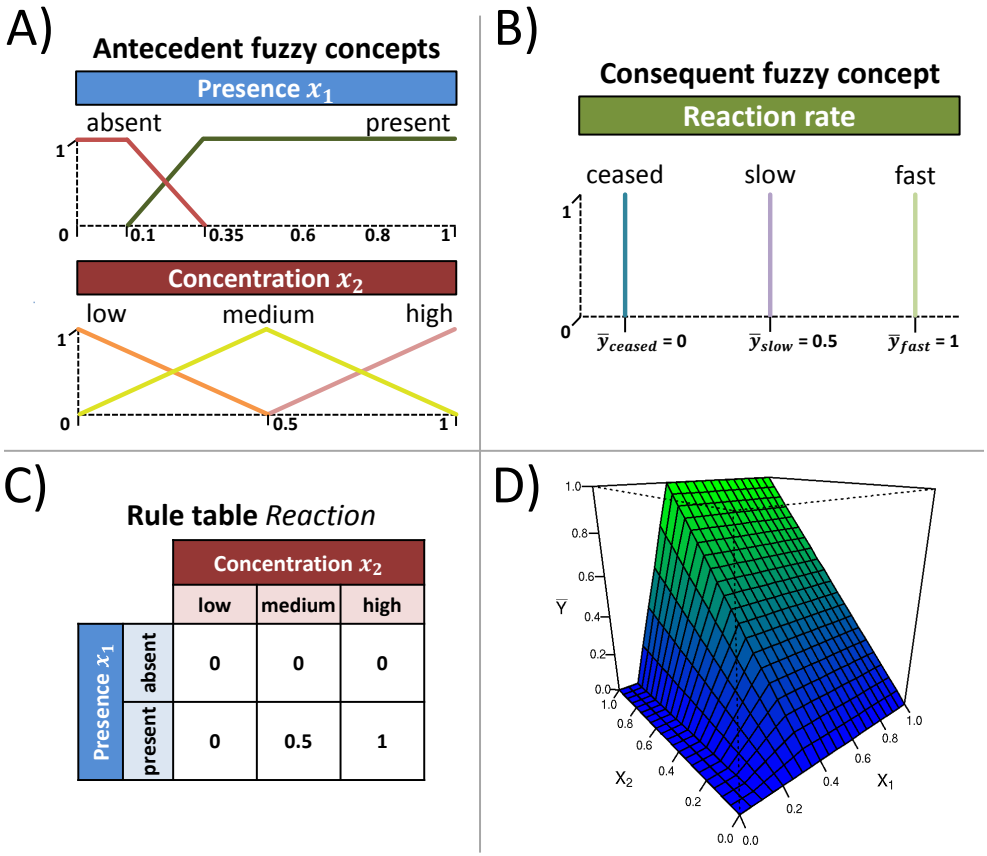


Figure 3.2 Example of a simple fuzzy logic system. A fuzzy logic system is created by defining antecedent fuzzy concepts for each input, a consequent fuzzy concept, and a rule base. **A)** In this example, the states of two input entities $x_1, x_2 \in [0,1]$ are fuzzified by two antecedent fuzzy concepts. Entity x_1 is described as being either *absent* or *present*, while entity x_2 is described as being present at a *low*, *medium*, or *high* concentration. **B)** The states of the two entities are mapped to a reaction rate, which is described by a consequent fuzzy concept. As we use height defuzzification, we only need to define the centers of gravity of antecedent fuzzy sets. Thus, we define these fuzzy sets as singleton fuzzy sets for simplicity. **C)** If-then rules can be represented by a rule table. From this rule table one can easily deduce that x_1 has to be present to allow a reaction, and that the reaction rate is proportional to the concentration of x_2. **D)** The fuzzy logic system maps input crisp values x_1 and x_2 to the output crisp value $\bar{y}$. The plot shows the image of $\bar{y}$, where colored faces illustrate the value of $\bar{y}$.

or equivalently by a rule table as in **Figure 3.2C**. For each rule the conclusion fuzzy set is derived using product inference:

$$\begin{aligned}
\mu_{B_1^*}(y) &= \mu_{absent}(x_1) \cdot \mu_{low}(x_2) \cdot \mu_{ceased}(y) = 0.2 \cdot 0.0 \cdot \mu_{ceased}(y) = 0.0 \cdot \mu_{ceased}(y) \\
\mu_{B_2^*}(y) &= \mu_{absent}(x_1) \cdot \mu_{medium}(x_2) \cdot \mu_{ceased}(y) = 0.2 \cdot 0.6 \cdot \mu_{ceased}(y) = 0.12 \cdot \mu_{ceased}(y) \\
\mu_{B_3^*}(y) &= \mu_{absent}(x_1) \cdot \mu_{high}(x_2) \cdot \mu_{ceased}(y) = 0.2 \cdot 0.4 \cdot \mu_{ceased}(y) = 0.08 \cdot \mu_{ceased}(y) \\
\mu_{B_4^*}(y) &= \mu_{present}(x_1) \cdot \mu_{low}(x_2) \cdot \mu_{ceased}(y) = 0.8 \cdot 0.0 \cdot \mu_{ceased}(y) = 0.0 \cdot \mu_{ceased}(y) \\
\mu_{B_5^*}(y) &= \mu_{present}(x_1) \cdot \mu_{medium}(x_2) \cdot \mu_{slow}(y) = 0.8 \cdot 0.6 \cdot \mu_{slow}(y) = 0.48 \cdot \mu_{slow}(y) \\
\mu_{B_6^*}(y) &= \mu_{present}(x_1) \cdot \mu_{high}(x_2) \cdot \mu_{fast}(y) = 0.8 \cdot 0.4 \cdot \mu_{fast}(y) = 0.32 \cdot \mu_{fast}(y)
\end{aligned}$$

The conclusion fuzzy value is derived by evaluating the membership values of the centers of gravity of conclusion fuzzy sets. Hereby, the center of gravity of a conclusion fuzzy set is identical to the center of gravity of the according consequent fuzzy set (Section A.3.3). Consequent fuzzy sets are typically designed such that the membership value of their centers of gravity equals 1. Thus, the entries of the conclusion fuzzy value can be seen as weights assigned to the consequent fuzzy sets:

$$\begin{aligned}
&< \mu_{B_1^*}(\bar{y}_{ceased}), \mu_{B_2^*}(\bar{y}_{ceased}), \mu_{B_3^*}(\bar{y}_{ceased}), \mu_{B_4^*}(\bar{y}_{ceased}), \mu_{B_5^*}(\bar{y}_{slow}), \mu_{B_6^*}(\bar{y}_{fast}) > \\
&=< 0.0,\ 0.12,\ 0.08,\ 0.0,\ 0.48,\ 0.32 >
\end{aligned}$$

By summing conclusion fuzzy value entries which correspond to the same consequent fuzzy set, the conclusion fuzzy value can be compacted:

$$\begin{aligned}
&< \mu_{B_1^*}(\bar{y}_{ceased}) + \mu_{B_2^*}(\bar{y}_{ceased}) + \mu_{B_3^*}(\bar{y}_{ceased}) + \mu_{B_4^*}(\bar{y}_{ceased}), \mu_{B_5^*}(\bar{y}_{slow}), \mu_{B_6^*}(\bar{y}_{fast}) > \\
&=< 0.2,\ 0.48,\ 0.32 >
\end{aligned}$$

Defuzzification of the (full) conclusion fuzzy value and defuzzification of the compacted conclusion fuzzy value results in the same crisp value:

$$\begin{aligned}
\bar{y} &= \frac{\sum_{j=1}^{6} \bar{y}_j \mu_{B_j^*}(\bar{y}_j)}{\sum_{j=1}^{6} \mu_{B_j^*}(\bar{y}_j)} \\
\Leftrightarrow \bar{y} &= \frac{\sum_{j=1}^{4} \bar{y}_{ceased} \mu_{B_j^*}(\bar{y}_{ceased}) + \bar{y}_{slow} \mu_{B_5^*}(\bar{y}_{slow}) + \bar{y}_{fast} \mu_{B_6^*}(\bar{y}_{fast})}{\sum_{j=1}^{6} \mu_{B_j^*}(\bar{y}_j)} \\
\Rightarrow \bar{y} &= \frac{0.0 \cdot 0.2 + 0.5 \cdot 0.48 + 1.0 \cdot 0.32}{0.2 + 0.48 + 0.32} = 0.56
\end{aligned}$$

Thus, the fuzzy logic system fls maps input crisp values $x_1 = 0.15$ and $x_2 = 0.7$ to output crisp value $\bar{y} = 0.56$. The image of fls is shown in **Figure 3.2D**. The qualitative behavior that we observe in the image becomes already apparent when studying the rule base and antecedent fuzzy sets of the fuzzy logic system. We observe in the image a linear increase of $\bar{y}$ with increasing x_2 if x_1 is present. This is reflected by the linear transition of x_2 from *low* to *medium* and from

medium to *high* as defined by the fuzzy set design, and by the rule base that assigns larger output values to higher x_2 concentrations. If x_1 is fully absent, we observe in the image that $\bar{y}$ is zero independent of the state of x_2. This behavior is apparent from the rule base. If x_1 is in the transition area between *absent* and *present*, the maximal $\bar{y}$ drops strongly in the image, but $\bar{y}$ still linearly increases with increasing x_2. This behavior can be interpreted as an intermediate between the two extreme behaviors that are defined by the rules for *absent* and *present* x_1.

3.2 Fuzzy Logic Systems Can Approximate Common Functions

To show that fuzzy logic systems are suited to model biological system, we demonstrate how fuzzy logic systems can be designed such that they are either identical to or approximate simple functions commonly used in other modeling techniques. In the following Section 4, we introduce the full Petri net and fuzzy logic framework and demonstrate how it can be used to simulate common biological motifs.

3.2.1 Hill and Michaelis-Menten Kinetics

The Hill equation can be used to describe binding of ligands to a receptor molecule, i.e. the Hill coefficient

$$\theta_{n,K_m}(L) = \frac{L^n}{(K_m)^n + L^n}$$

quantifies the fraction of receptor-sites bound by ligands. Hereby, L is the concentration of ligands, K_m the ligand concentration producing half occupation, and n controls the shape of the sigmoid curve. If $n = 1$ Hill kinetics reduce to the well know Michaelis-Menten kinetics used to describe the rate of enzymatic reactions. Ligand-bound receptors typically exhibit a certain functionality in biological systems, e.g. by catalyzing the phosphorylation of downstream signaling molecules. In an ODE model, the rate of these downstream reactions can be quantified as $\frac{dy}{dt} = V_{max} \cdot \theta_{n,K_m}(L)$ for activating effects, or $\frac{dy}{dt} = V_{max} \cdot (1 - \theta_{n,K_m}(L))$ for inhibiting effects, where V_{max} is the maximal rate of the reaction. Please note that $(1 - \theta_{n,K_m}(L)) = \theta_{-n,K_m}(L)$ holds.

Such a functionality can be easily approximated by fuzzy logic systems (**Figure 3.3**). To design a fuzzy logic system that should approximate a sigmoid function $f_{sig} : X \to Y$, use the following procedure:

1. Depending on the shape of f_{sig}, restrict to a reasonable range of the domain of discourse. States within this range will be described in detail by fuzzy sets. States outside this range will be covered by unbounded fuzzy sets only, i.e. one assumes that the exact values of such states do not matter. As an example, we restrict to the interval $[0,1]$.
2. Choose the number of fuzzy sets that are used to discretize the chosen range of the domain of discourse. In general, the more fuzzy sets are used, the more complex is the final fuzzy logic system and the closer is the approximation. Here, we discretize the interval $[0,1]$ using four fuzzy sets.

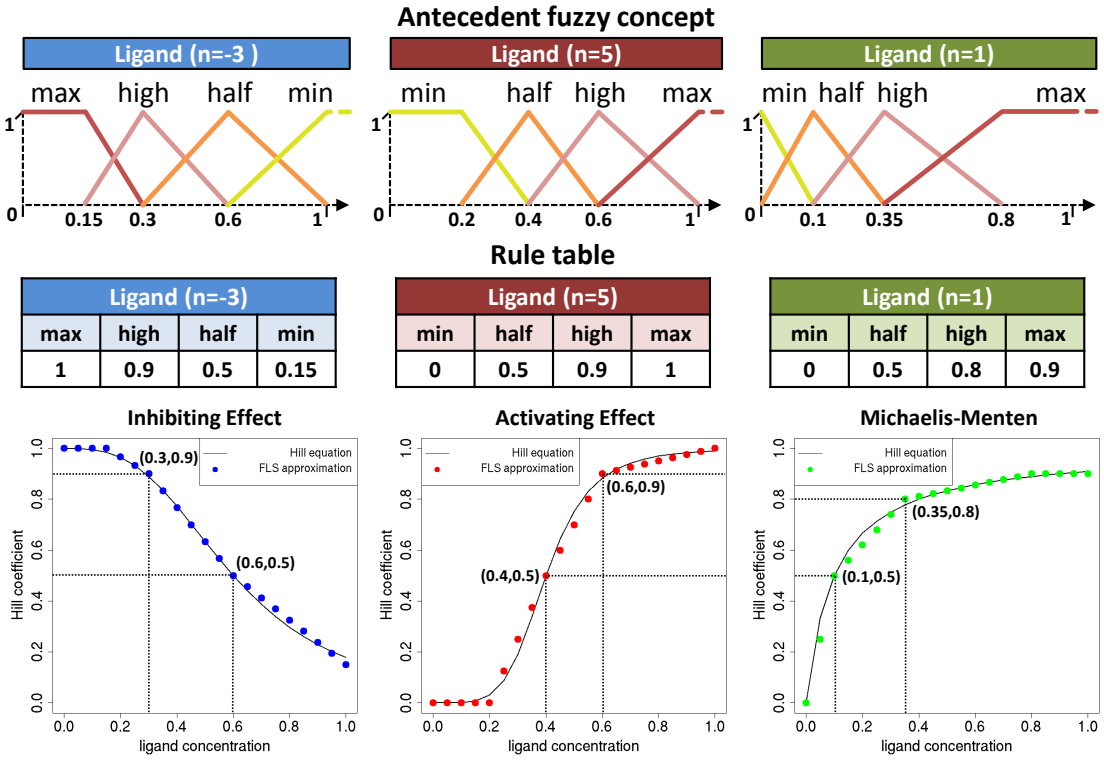


Figure 3.3 Fuzzy logic systems approximate Hill and Michaelis-Menten functions. Fuzzy logic systems can be easily designed such that they approximate the sigmoid-shaped images of Hill functions describing: inhibitory effects (left), activating effects (center), or Michaelis-Menten kinetics (right). Using four fuzzy sets (top) and simple rule bases (middle), one creates piecewise linear approximations of Hill functions for $L \in [0,1]$ with n=(-3,5,1) and K_m =(0.6,0.4,0.1) (left,center,right). See main text for a design procedure. The approximation quality depends on the number of fuzzy sets, but is already quite good if only four fuzzy sets are used (compare lines and dots in the bottom plots). The tuples specify examples for $(x, fls(x))$, where x is a ligand concentration and $fls(x)$ the fuzzy logic systems approximating the Hill function.

3. Choose an according number of pairs $(x_j, f_{sig}(x_j)) \in X \times Y$. Here, that would be pairs of (ligand concentration, Hill coefficient), e.g. (0.0,1.0), (0.3,0.9), (0.6,0.5), and (1.0,0.15) (**Figure 3.3**, left).
4. Sort pairs ascending according to the value of x_j. For each pair $(x_j, f_{sig}(x_j))$:
   (a) Add a triangular fuzzy set μ_{X_j} with $l = x_{j-1}$, $r = x_{j+1}$, and $mp = x_j$ to the antecedent fuzzy concept (Section 2.1). Use unbounded fuzzy sets as leftmost and rightmost fuzzy sets.
   (b) Add a singleton fuzzy set μ_{Y_j} with center of gravity $\bar{y}_j = f_{sig}(x_j)$ to the consequent fuzzy concept.
   (c) Add a rule to the rule base that maps the newly created antecedent fuzzy set to the newly created consequent fuzzy set.

Proposition The resulting fuzzy logic system is a piecewise linear approximation of f_{sig}.

Proof Let $fls : X \to Y$ be a fuzzy logic system created by the aforementioned procedure. This fuzzy logic system has a single input crisp value which is described by the antecedent fuzzy concept $< \mu_{X_1}, \ldots, \mu_{X_m} >$. Per definition, the contained fuzzy sets are defined as following:

$$\mu_{X_1}(x) = \begin{cases} 1 & \text{if } x \leq x_1 \\ \frac{x_2 - x}{x_2 - x_1} & \text{if } x_1 < x < x_2 \\ 0 & \text{if } x_2 \leq x \end{cases} \qquad \mu_{X_j}(x) = \begin{cases} 0 & \text{if } x \leq x_{j-1} \\ \frac{x - x_{j-1}}{x_j - x_{j-1}} & \text{if } x_{j-1} < x < x_j \\ 1 & \text{if } x = x_j \\ \frac{x_{j+1} - x}{x_{j+1} - x_j} & \text{if } x_j < x < x_{j+1} \\ 0 & \text{if } x_{j+1} \leq x \end{cases}$$

$$\mu_{X_m}(x) = \begin{cases} 0 & \text{if } x \leq x_{m-1} \\ \frac{x - x_{m-1}}{x_m - x_{m-1}} & \text{if } x_{m-1} < x < x_m \\ 1 & \text{if } x_m \leq x \end{cases} \qquad \text{where } 1 < j < m.$$

Per definition, the fuzzy logic system has a consequent fuzzy concept $< \mu_{Y_1}, \ldots, \mu_{Y_m} >$ with centers of gravity $(\bar{y}_1, \ldots, \bar{y}_m)$. The rule base maps antecedent fuzzy set μ_{X_i} to consequent fuzzy set μ_{Y_i}. The fuzzy logic system can be written as:

$$fls(x) = \bar{y} = \frac{\sum_{j=1}^{m} \bar{y}_j \mu_{Y_j^*}(\bar{y}_j)}{\sum_{j=1}^{m} \mu_{Y_j^*}(\bar{y}_j)} = \frac{\sum_{j=1}^{m} \bar{y}_j \cdot \mu_{X_j}(x)}{\sum_{j=1}^{m} \mu_{X_j}(x)}$$

We consider three cases with respect to the value of x:

Case 1: Let $x < x_1$. Then $fls(x)$ reduces to

$$fls(x) = \frac{y_1 \cdot \mu_{X_1}(x)}{\mu_{X_1}(x)} = y_1$$

Case 2: Let $x > x_m$. Then $fls(x)$ reduces to

$$fls(x) = \frac{y_m \cdot \mu_{X_m}(x)}{\mu_{X_m}(x)} = y_m$$

Case 3: Let $x_1 \leq x \leq x_m$. Then obviously $x_{j-1} \leq x \leq x_j$ for some j. Then $fls(x)$ reduces to

$$fls(x) = \frac{y_{j-1} \cdot \mu_{X_{j-1}}(x) + y_j \cdot \mu_{X_j}(x)}{\mu_{X_{j-1}}(x) + \mu_{X_j}(x)} = \frac{y_{j-1} \cdot \frac{x_j - x}{x_j - x_{j-1}} + y_j \cdot \frac{x - x_{j-1}}{x_j - x_{j-1}}}{\frac{x_j - x}{x_j - x_{j-1}} + \frac{x - x_{j-1}}{x_j - x_{j-1}}}$$

$$\Leftrightarrow fls(x) = \frac{y_{j-1} \cdot (x_j - x) + y_j \cdot (x - x_{j-1})}{(x_j - x) + (x - x_{j-1})} = \frac{y_{j-1}x_j - y_{j-1}x + y_j x - y_j x_{j-1}}{x_j - x_{j-1}}$$

$$\Leftrightarrow fls(x) = \frac{(y_j - y_{j-1})}{x_j - x_{j-1}} x + \frac{y_{j-1}x_j - y_j x_{j-1}}{x_j - x_{j-1}} = c_a \cdot x + c_b$$

with constant slope c_a and intercept c_b. In all three cases, $fls(x)$ is a linear function. Thus, the fuzzy logic system is a piecewise linear function. As $fls(x_j) = f_{sig}(x_j)$ for $j \in \{1, \ldots, m\}$, the fuzzy logic system approximates the sigmoidal function in $[x_1, x_m]$, **q.e.d.**

3.2.2 Mass-Action Kinetics

Functions representing mass-action kinetics are very common in ODE modeling (Section 1.1). Hereby, the current states of entities are represented by their concentrations as (crisp) real values. The state-change of a target entity depends linearly on the states of one or more input entities, modified by a kinetic constant:

$$\frac{dy}{dt} = k \cdot x_1 \cdot \ldots \cdot x_n = k \cdot \prod_{i=1}^{n} x_i \tag{3.3}$$

Proposition If the domain of discourse of each x_i is $[0,1]$, i.e. when relative concentrations are considered, then Equation 3.3 can be stated as a fuzzy logic system whose defuzzified output value $\bar{y}$ equals $\frac{dy}{dt}$.

Proof Each concentration x_i is represented by two triangular fuzzy sets μ_{low} and μ_{high}, such that $\mu_{low}(x_i) = 1 - x_i$ and $\mu_{high}(x_i) = x_i$ holds (**Figure 3.4**). The conclusion fuzzy sets are $\mu_{zeroRate}$ and $\mu_{maxRate}$, with the following centers of gravity:

$$\bar{y}_{zeroRate} = 0 \qquad\qquad \bar{y}_{maxRate} = k$$

The rule base consists of 2^n rules, i.e. the antecedents consist of all possible combinations of fuzzifications for the n input crisp values. Let the first rule specify that all input crisp values are fuzzified using fuzzy set μ_{high} and that these are mapped to the consequent fuzzy set $\mu_{maxRate}$:

$$R_1 : IF\ x_1\ is\ high\ AND\ x_2\ is\ high\ AND\ \ldots\ AND\ x_n\ is\ high\ THEN\ y\ is\ maxRate$$

The other $2^n - 1$ rules, where at least one x_i is fuzzified by fuzzy set μ_{low}, map the inputs to the consequent fuzzy set $\mu_{zeroRate}$ with $\bar{y}_{zeroRate} = 0$. Thus, using the compact notion of a fuzzy logic

system from Equation 3.2 the resulting fuzzy logic system can be simplified:

$$\bar{y} = fls(x_1,\ldots,x_n) = \frac{\sum_{j=1}^{2^n} \bar{y}_j \cdot \prod_{i=1}^{n} \mu_{A_{j,i}}(x_i)}{\sum_{j=1}^{2^n} \prod_{i=1}^{n} \mu_{A_{j,i}}(x_i)} = \\ = \frac{\bar{y}_{maxRate} \cdot \prod_{i=1}^{n} \mu_{high}(x_i)}{\sum_{j=1}^{2^n} \prod_{i=1}^{n} \mu_{A_{j,i}}(x_i)} \tag{3.4}$$

As the antecedents of the rule base consist of all possible combinations of fuzzy sets for the n input crisp values, the denominator of Equation 3.4 can be rewritten and reorganized:

$$\sum_{j=1}^{2^n} \prod_{i=1}^{n} \mu_{A_{j,i}}(x_i) = \sum_{s_1 \in \{low,high\}} \sum_{s_2 \in \{low,high\}} \cdots \sum_{s_n \in \{low,high\}} \prod_{i=1}^{n} \mu_{s_i}(x_i) = \\ = \sum_{s_1 \in \{low,high\}} \mu_{s_1}(x_1) \sum_{s_2 \in \{low,high\}} \mu_{s_2}(x_2) \ldots \sum_{s_n \in \{low,high\}} \mu_{s_n}(x_n)$$

As $\forall i \in \{1,\ldots,n\} : \sum_{s_i \in \{low,high\}} \mu_{s_i}(x_i) = \mu_{low}(x_i) + \mu_{high}(x_i) = 1 - x_i + x_i = 1$, the denominator of Equation 3.4 equals 1. Thus, Equation 3.4 can be reduced to

$$\bar{y} = fls(x_1,\ldots,x_n) = \bar{y}_{maxRate} \cdot \prod_{i=1}^{n} \mu_{high}(x_i) = \bar{y}_{maxRate} \cdot \prod_{i=1}^{n} x_i = k \cdot \prod_{i=1}^{n} x_i$$

which is identical to the right hand side of Equation 3.3. Thus, the defuzzified output value $\bar{y}$ of the fuzzy logic system is identical to $\frac{dy}{dt}$ of the mass-action kinetics function of the ODE model, **q.e.d.**

The conversion of absolute to relative concentrations is straightforward, including the according adaption of kinetic constants. Hereby, each x_i is dived by a sufficiently high real number and the kinetic constant k is multiplied by the same numbers. Thus, fuzzy logic systems are able to mimic mass-action kinetics if concentrations are bounded. See **Figure 3.4** for an example.

3.2.3 Logical Gates

When representing the effects of different transcription factors binding to the promotor of a gene and influencing its expression, functions similar to logic disjunctions or conjunctions are commonly used (OR-like and AND-like functions). An OR-like function would represent the effect of a transcription factor that can influence gene expression independent of the respective other transcription factor, i.e. binding of one of the two transcription factors is sufficient to allow gene expression. An AND-like function would represent the effect of dependent transcription factors, i.e. both transcription factors have to be bound such that gene expression is performed. In a Boolean model, such a functionality could be realized by OR and AND gates defined by the following rule tables:

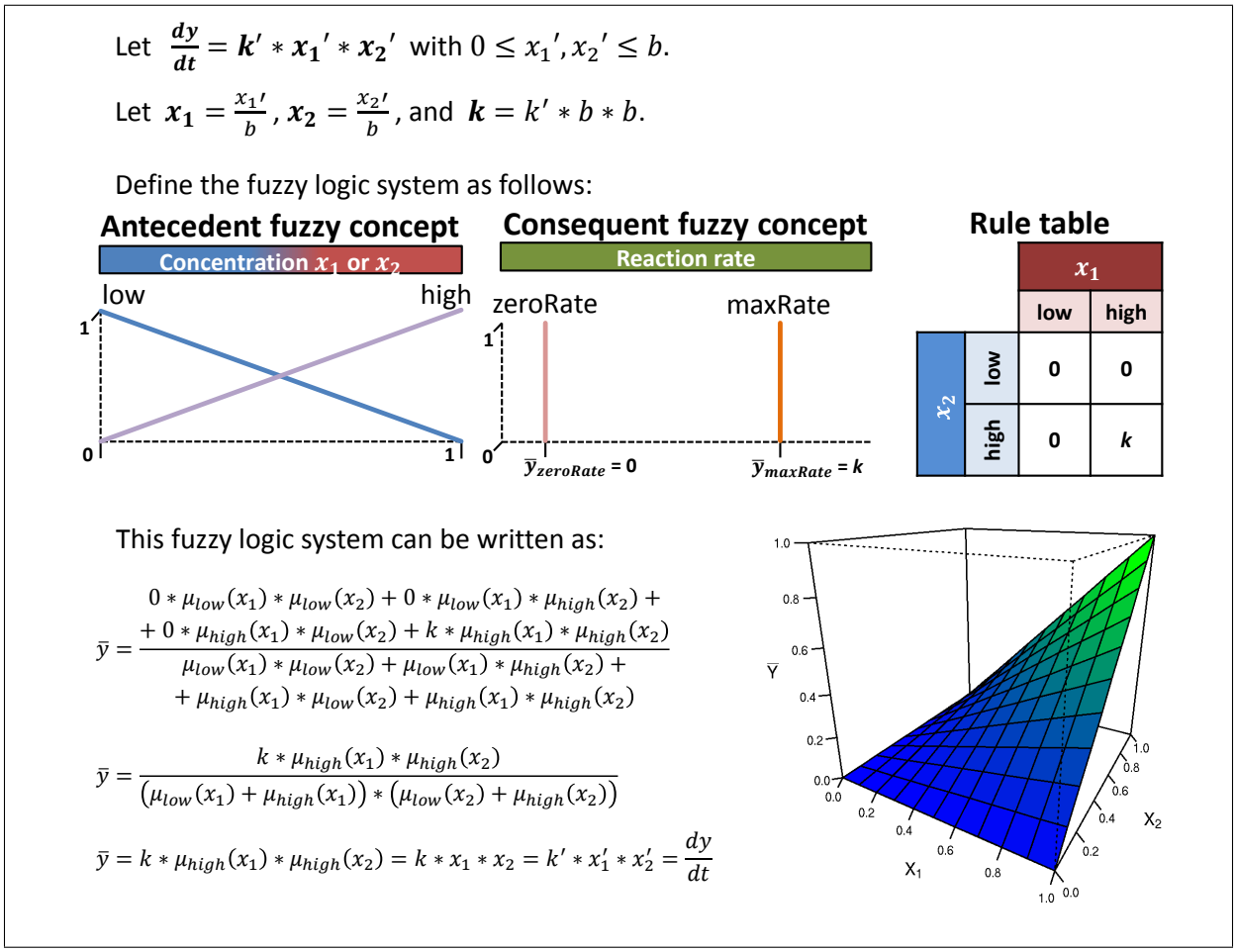


Figure 3.4 Fuzzy logic systems can mimic mass-action functions. Mass-action functions as used in ODE models can be stated as fuzzy logic systems. This is proven in the main text. Here, we give an example of a fuzzy logic system which mimics a mass-action function with two factors x_1 and x_2. The only necessary condition is that the values of x_1 and x_2 are bounded by a known value b. The plot shows the image of $\bar{y}$ (bottom right). Colored faces illustrate the value of $\bar{y}$.

OR gate		TF_1	
		0	1
TF_2	0	0	1
	1	1	1

AND gate		TF_1	
		0	1
TF_2	0	0	0
	1	0	1

Similar rule tables can be used in fuzzy logic systems that mimic OR and AND gates. Hereby, the states of transcription factors are represented by suitable fuzzy sets. In the simplest case by two fuzzy sets (**Figure 3.5** left) or when considering more complicated effects by four fuzzy sets (**Figure 3.5** right) and appropriate consequent fuzzy sets. The appropriate fuzzy logic systems exhibit the same qualitative behavior, but have a smooth transition from "falsehood" (0) to "truth" (1) due to the fuzzy set design. Notice that fuzzy sets can be designed such that the transition between *absent* and *present* is abrupt, i.e. that the right border and right top of *absent* and left border and left top of *present* are identical, e.g. at 0.5. In such a case, the smooth transition disappears and the fuzzy logic system would exhibit a switch-like behavior resulting in either $\bar{y} = 0$ or $\bar{y} = 1$, but no intermediate values. Thereby, the fuzzy logic system reduces to a classical OR or AND gate.

3.3 Discussion

Fuzzy logic systems have a one-to-one correspondence to linguistic descriptions of the represented processes. Linguistic descriptions of interactions between entities can be converted into the rule bases that are a constituting part of each fuzzy logic systems. *Vice versa*, each rule base is a linguistic description of the functionality of the fuzzy logic system and therefore a description of the represented biological process. The rule bases can easily and straightforwardly be formulated and can easily be understood by a user, even if this user is not particularly familiar with mathematical formulations. The rules allow to interpret the behavior of a function in dependence to each input entity's state, which in turn can be interpreted based on its fuzzy set description (see example in Section 3.1).

The fuzzy set based descriptions are used as antecedents for fuzzy logic systems. Different descriptions for the same property of an entity can be used in different fuzzy logic systems. Thus, there is not necessarily only one description of concentration, presence, activity, etc of an entity. As we have mentioned in the previous Section 2, functional considerations influence fuzzy set design. As an entity may be part of several processes and may affect each differently, a process and thus fuzzy logic system specific representation is often meaningful.

A fuzzy logic system uses crisp (real valued) numbers as inputs, fuzzifies, performs approximate reasoning, defuzzifies, and finally creates a crisp number as output. Thus, the fuzzy set based representation by fuzzy values is an inherent part of the reasoning process, but this process itself uses and creates crisp number based representations of states. PNFL models, which will be introduced in the following Section 4, store current states of entities as crisp numbers, which are only converted to a fuzzy set based representation during the reasoning process, i.e. during fuzzy logic system evaluation.

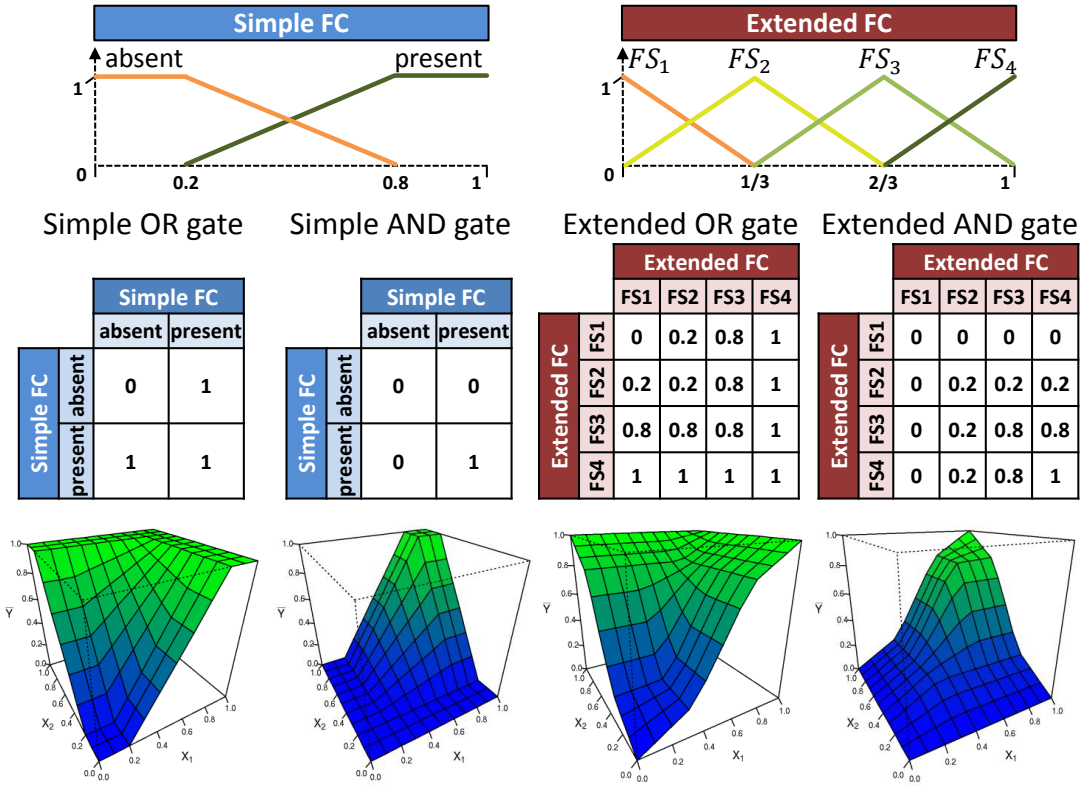


Figure 3.5 Fuzzy logic systems can mimic logic gates. Logic gates are used in Boolean models to represent cooperative effects (OR-like and AND-like functions). Such functions can be implemented by fuzzy logic systems. Depending on the number of antecedent fuzzy sets (top), such fuzzy logic systems reproduce a simple (left) or more complex behavior (right), representing e.g. a sigmoid-like response. The rule bases are defined analogously to logic gates (center). The plots show the images of the fuzzy logic systems (bottom). Colored faces illustrate the value of $\bar{y}$.

A comparison to ODE and discrete logic models In ODE models, virtually all imaginable functions can be included to describe the processes of the modeled system. However, many ODE models use a selection of simple functions for representing pairwise interactions [24, 78]. More complex functions typically only subsume and approximate several simple processes, for example Michaelis-Menten kinetics that are used to calculate enzymatic reaction rates based on substrate concentrations only. Hereby, the processes of enzyme-substrate complex formation, catalyzation, complex dissolving, and the according reverse processes, which all follow linear mass-action kinetics, are approximated by the well known nonlinear function. Thus, most cellular processes can be represented by simple functions.

Section 3.2.2 demonstrates that despite the fuzzy discretization of input states, fuzzy logic systems are capable to mimic mass-action kinetics and thus can be used to represent common linear and concentration dependent reactions. Furthermore, Section 3.2.1 demonstrates that fuzzy logic systems can be designed such that they approximate common non-linear kinetics, namely kinetics based on the Hill equation with arbitrary parameters. Thus, fuzzy logic systems can mimic or approximate complex functionalities of ODE models, while their rule bases can be directly and easily interpreted.

The functions used in discrete logic models are based on rule tables that map discrete states of input entities to a discrete state of the output entity. The rule bases of fuzzy logic system correspond to this rule tables and may provide a similar functionality (Section 3.2.3). Predictions of discrete logic functions are restricted to the predefined discrete state. In contrast, fuzzy logic systems extend this functionality by allowing smooth transitions between states. This shows the modeling power that fuzzy logic systems provide together with straightforward interpretability.

Author's contribution Fuzzy logic systems are an established concept [107]. The author describes various aspects of their application to represent biological data and shows that fuzzy logic systems can mimic common functionalities of ODE and discrete logic models.

Chapter 4

Joining Petri Nets and Fuzzy Logic: PNFL Modeling

In the previous sections we have demonstrated how fuzzy sets represent states of entities and how they are utilized within fuzzy logic systems to describe antecedents and consequents of rules. We have seen that fuzzy logic systems specify the functionality of interactions and that they can mimic common functions like sigmoids or mass-action kinetics.

In this section, we introduce the Petri net and fuzzy logic (PNFL) modeling framework for biological systems. We combine fuzzy sets and fuzzy logic systems with Petri nets, which are used to define the connectivity of networks and serve as graphical representations.

First, we give a brief formal definition of PNFL models. Second, we present several design guidelines for building PNFL models. This includes modeling of processes involving multiple effectors, and ODE- and Boolean-like modeling approaches. Third, we provide PNFL models for four common biological motifs as application examples. These motifs can be seen as small biological systems. We demonstrate how PNFL models can be designed to achieve the well known functionality of the motifs, what kinds of analysis can be performed based on PNFL models, and what effects the design of fuzzy sets has on the quantitative and qualitative behavior of the models.

4.1 Definition of a PNFL Model

A *Petri net with fuzzy logic* (PNFL) is an instance of a hybrid functional Petri net [108] defined as a 6-tuple $PN = (P, T, A, F, W, M_0)$ where

$P = \{p_1, p_2, \ldots, p_m\}$	is a finite set of places,
$T = \{t_1, t_2, \ldots, t_n\}$	is a finite set of transitions,
$A \subseteq (P \times T \cup T \times P)$	is a set of arcs,
$F = \{f_0, f_1, \ldots, f_s\}$	is a finite set of functions $f_i : \mathbb{R} \times \ldots \times \mathbb{R} \to \mathbb{R}$
$W : A \to F$	is a function that assigns a single $f_i \in F$ to each arc,
$M_0 : P \to \mathbb{R}$	is the initial marking

with $P \cap T = \varnothing$. See **Figure 4.1** for an exemplary PNFL model.

Each place represents a single property of a (biological) entity, for example, the concentration of a certain protein species or the fold-change of a certain mRNA species. If multiple properties of the same biological entity are of interest for modeling, then multiple places are used for the representation. For example, a place represents the concentration of a protein species while another place represents the relative abundance of its mature form.

The current state of an entity with respect to a property is defined by the current marking of the respective place. The marking of a place p at time k is a crisp number $M_k(p) \in \mathbb{R}$. We assume the domain of real values as the domain of discourse. Whenever we write about the value or state of a place, we refer to the marking of this place.

Transitions represent arbitrary (biological) processes. For example, an enzymatic reaction, translation of a certain mRNA, or regulation of a gene's expression. A process depends on the current states of some entities and causes state changes of others. An arc from a place to a transition (input arc) represents that the place affects the according process, while an arc from transition to place (output arc) represents that the entity is affected by the process.

A function assigned to an output arc (t_j, p_i) defines the value added to the current marking of p_i whenever transition t_j fires, while a function assigned to an input arc (p_i, t_j) defines the value that is consumed from the current marking of p_i. The markings of all places connected to a transition can be used as arguments for the functions assigned to the arcs.

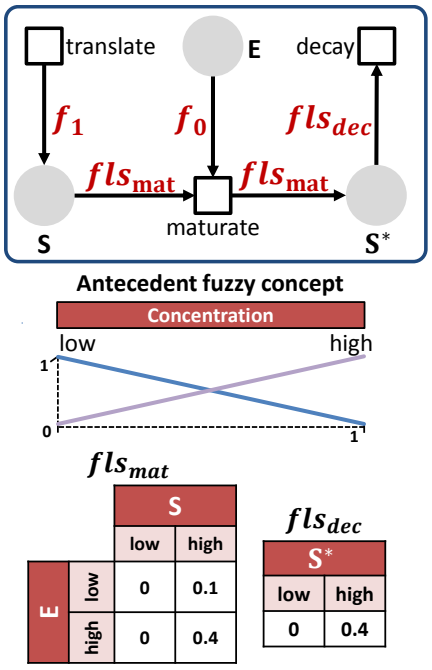
In most cases these functions are fuzzy logic systems, but arbitrary functions are possible. For example, the zero function $f_0 = 0$ is often assigned to input arcs. It specifies that the marking of the adjacent place is not changed during firing, while it can still be used in other functions of arcs connected to the same transition. Such arcs are often called *read arcs* or *test arcs*. Read arcs reflect the fact that many processes are influenced by entities which are themselves not affected by the process, e.g. the pH-value of the cellular environment can be modeled as a place. It may affect the conversion rate of an enzyme while the pH-value itself is not affected by the enzymatic reaction.

4.1.1 Simulation of PNFL Models

Simulation of a model means computing state changes for all entities during a predefined time interval starting from some initial states. In PNFL models, state changes are realized by firing of transitions. If a transition fires, the functions assigned to input and output arcs are evaluated. Immediately after function evaluation, the values of the input and output places are updated, i.e. the current marking M_k of these places is updated to M_{k+1}. For example, we consider a transition t and a place p with an input arc (p,t) as well as an output arc (t,p). The marking of p would be updated as follows:

$$M_{k+1}(p) = M_k(p) + f_{(t,p)}(\ldots) - f_{(p,t)}(\ldots)$$

where $f_{(t,p)}(\ldots)$ is the function assigned to the output arc from t to p and $f_{(p,t)}(\ldots)$ is the function assigned to the input arc from p to t. Remember that output arcs add and input arcs subtract the function values from the marking of the respective place. The elapsed time is quantified by the



$$P = \{S, S^*, E\}$$
$$T = \{translate, maturate, decay\}$$
$$A = \{(translate, S), (S, maturate), (E, maturate), (maturate, S^*), (S^*, decay)\}$$
$$F = \{f_0 = 0, f_1 = 0.1, fls_{mat}(S,E) = (defined\ left), fls_{dec}(S^*) = (defined\ left)\}$$
$$W(a) = \begin{cases} f_0 & \text{if } a = (E, maturate) \\ f_1 & \text{if } a = (translate, S) \\ fls_{mat}(S,E) & \text{if } a = (S, maturate) \\ fls_{mat}(S,E) & \text{if } a = (maturate, S^*) \\ fls_{dec}(S^*) & \text{if } a = (S^*, decay) \end{cases}$$
$$M_0(S) = 1.0 \quad M_0(S^*) = 0.0 \quad M_0(E) = 0.8$$

Figure 4.1 Example of a PNFL model. A PNFL model is defined by a set of places P, a set of transitions T, a set of arcs A connecting places to transitions and *vice versa*, a set of functions F, an assignment of functions to arcs $W : A \to F$, and an initial marking M_0. A place p represents a property of a (biological) entity. Its state at time k is given by the marking $M_k(p)$. If a transition t fires, the functions assigned to incident arcs are evaluated. Output arcs (t,p) add, input arcs (p,t) subtract the function values from the current marking of p. Functions may be arbitrary to allow for flexibility of models, but most often include fuzzy logic systems. This simple PNFL model represents the concentration of a protein substrate as S, the concentration of its mature form as S^*, and the concentration of an enzyme as E. The translation reaction occurs at a constant rate ($f_1 = 0.1$). The maturation reaction of S to S^* is represented by an enzyme-dependent fuzzy logic system ($fls_{mat}(S,E)$), the decay of S^* mimics an exponential decay ($fls_{dec}(S^*)$). The enzyme's concentration E is not influenced by the maturation process ($f_0 = 0$).

number of firing iterations, i.e. the number of updates of the marking. The series of markings $M_0, \ldots, M_k$ provides time courses for the states of all entities of the system. The order of firing is determined by firing rules, i.e. a firing rule determines, which transitions are fired next. Examples for firing rules are:

Random firing: Transitions are randomly chosen, one by one. The marking is updated after each firing of a transition. The resulting simulation is non-deterministic and stochastic effects might significantly change the resulting time courses.

Stochastic firing: Transitions are preferably chosen according to the marking of adjacent places, e.g. using the Gillespie algorithm [109]. Transitions fire one by one. The marking is updated after each firing of a transition. The resulting simulation is non-deterministic.

Simultaneous firing: All transitions fire simultaneously. The marking is updated after all transitions have fired. If multiple transitions fire simultaneously, all according functions are evaluated based on M_k, which is then updated to M_{k+1}.

$$M_{k+1}(p) = M_k(p) + \sum_{t \in {}^\bullet(p)} f_{(t,p)}(\ldots) - \sum_{t \in (p)^\bullet} f_{(p,t)}(\ldots)$$

where the first sum iterates over all transitions with output arcs to p and the second sum iterates over all transition with input arcs from p. The resulting simulation is deterministic and resulting time courses are reproducible.

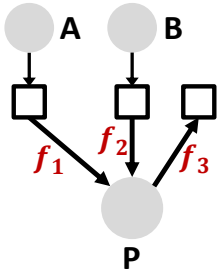
If not stated differently, simultaneous firing for the simulation of PNFL models is implied throughout this document.

4.1.2 Modeling Multiple Effectors

An entity of a system can be part of multiple processes that alter its state, and each of these processes can be affected by several entities including the target entity. Such entities are denoted as effectors. For example, the expression level of a mRNA species could be affected by the concentration of several transcription factors. Additionally, the mRNA is subject to decay and therefore is its own effector. Each effector has an individual effect on its target entity. An effect is quantified by evaluating functions that are assigned to arcs.

Effects are mutually independent if they can be independently evaluated and additively integrated. I.e. the function value of one effect is not used as an input to another function which evaluates another effect, and all effects can be independently added to or subtracted from the current state of the target entity. Mutually independent effects can be easily implemented in PNFL models. Each of those effects can be modeled as an individual transition with according arcs (**Figure 4.2**).

It is emphasized that although effects may be mutually independent, they can be part of the same biological process, e.g. the effects of several transcription factors can be independent and additive, but they still affect the same transcription process. Thus, multiple transitions could be used to model the same biological process if this is reasonable.



$$M_{k+1}(P) = M_k(P) + f_1(A) + f_2(B,C) - f_3(C)$$

Figure 4.2 Mutually independent effects. The state of an entity C is affected by several effectors: A, B and itself. The individual effects are mutually independent and quantified by evaluating functions $f_1(A)$, $f_2(B,C)$, and $f_3(C)$. If this PNFL model is simulated using simultaneous firing, the effects are jointly added or subtracted from the current marking $M_k(C)$ to get the next marking $M_{k+1}(C)$.

If mutually dependent effectors exert an effect on a target entity, their individual non-additive effects have to be integrated. A single transition and an according arc are used to represent the joint effect, and the function assigned to the arc is used to quantify it. Fuzzy logic systems can be designed for this task basically in two ways. Either a single fuzzy logic system with multiple antecedents and a multidimensional rule table is used, or the defuzzified output values of multiple fuzzy logic systems with few inputs are totalized (**Figure 4.3**).

A fuzzy logic system with a multidimensional rule table allows the formulation of very complex functions. If all possible combinations of antecedent fuzzy sets should be covered by a rule, the number of rules increases exponentially depending on the number of antecedents. For example, a fuzzy logic system with two antecedents each fuzzified using three fuzzy sets will have $3^2 = 9$ rules, with three antecedents $3^3 = 27$ rules, with four antecedents $3^4 = 81$ rules, etc. Thus, definition and interpretation is only feasible for a small number of antecedents.

Independent evaluation of simple fuzzy logic systems and totalizing their defuzzified output values avoids this curse of dimensionality. The trade-off is that the potential complexity of the resulting function is reduced. A convenient way of totalizing is to calculate a generalized mean of defuzzified output values. This allows a simple representation of cooperative effects:

$$GM_p(\bar{y}_1,\ldots,\bar{y}_n) = \left(\frac{1}{n}\sum_{i=1}^{n}\bar{y}_i^p\right)^{\frac{1}{p}}$$

where parameter p controls the type of cooperative effect:

$$\begin{aligned}
p \to -\infty: &\quad GM_p \mathrel{\hat{=}} \text{minimum}\\
p < 1: &\quad GM_p \mathrel{\hat{=}} \text{AND-like}\\
p = 1: &\quad GM_p \mathrel{\hat{=}} \text{arithmetic mean}\\
p > 1: &\quad GM_p \mathrel{\hat{=}} \text{OR-like}\\
p \to \infty: &\quad GM_p \mathrel{\hat{=}} \text{maximum}
\end{aligned}$$

Several GM_p functions can be nested to represent more complex dependencies, e.g. $f(A,B,C) = (A \text{ AND } B) \text{ OR } C$ can be approximated by $f(A,B,C) = GM_5(GM_{-5}(A,B),C)$.

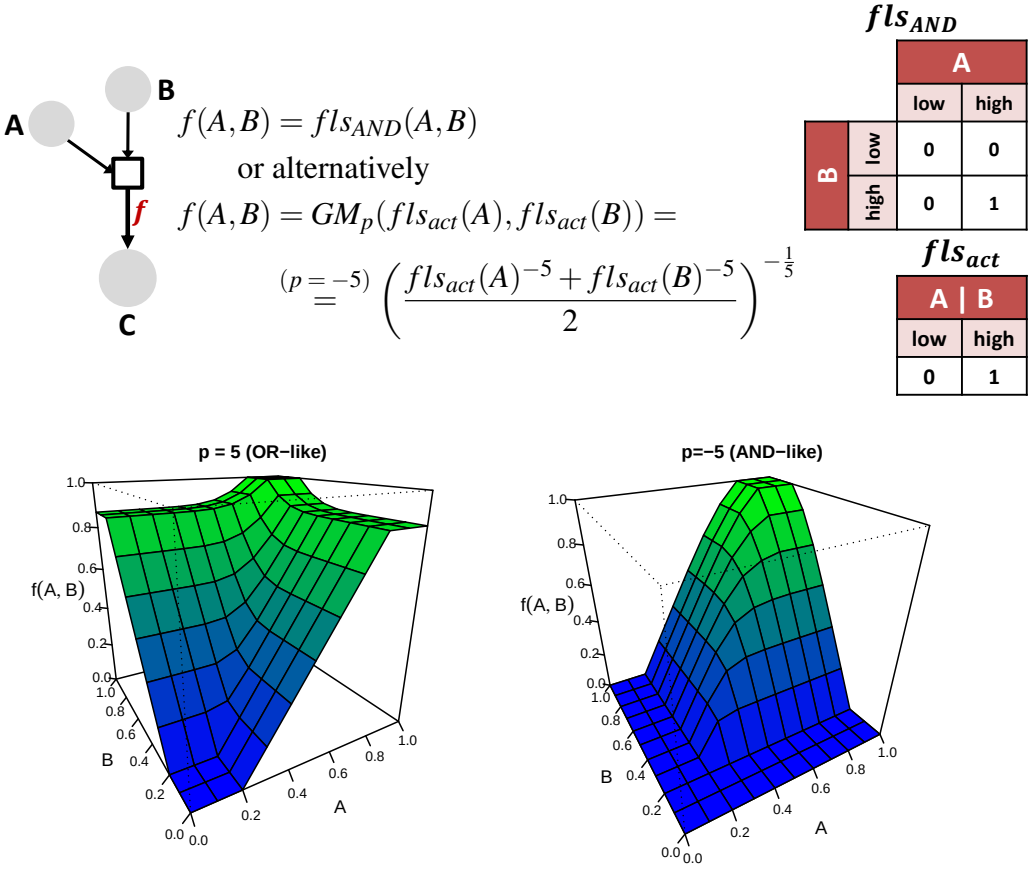


Figure 4.3 Integration of multiple effects. Mutually dependent effects of multiple effectors can be integrated using multidimensional fuzzy logic systems, e.g. $fls_{AND}(A,B)$, or alternatively by totalizing the outputs of multiple, simple fuzzy logic systems, e.g. by using the generalized mean $GM_p(fls_{act}(A), fls_{act}(B))$. Totalizing simple fuzzy logic systems helps to avoid the curse of dimensionality, i.e. keeps the number of necessary rules small. Generalized means can be used to approximate AND- or OR-like dependencies. The plots at the bottom of this figure show the images of $GM_5(A,b)$ (left, OR-like) and $GM_{-5}(A,B)$ (right, AND-like) for $A,B \in [0,1]$.

4.1.3 Semi-Continuous and Semi-Discrete Modeling

A computational modeling framework should allow the user to implement her knowledge and hypotheses about the effects between entities of a biological system as convenient as possible. This facilitates model creation, especially for laymen. There are two principal possibilities how knowledge about effects can be structured and described.

First, change-oriented descriptions in the form of: The current state of a group of entities induces a certain change of the state of a target entity. Or more formal: If x_1 is A_1 and x_2 is A_2 then Δy is B. For example:

- If the concentration of the transcription factor is high, then the expression level of the mRNA increases fast.
- If the concentration of the transcription factor is low, then the expression level of the mRNA increases slowly.

Ordinary differential equation (ODE) models are suited to directly implement this type of knowledge. The differential equations specify state changes depending on current states.

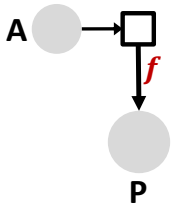
Second, state-oriented descriptions in the form of: The current state of a group of entities causes that a target entity adopts a certain state. Or more formal: If x_1 is A_1 and x_2 is A_2 then y is B. For example:

- If the concentration of the transcription factor is high, then the expression level of the mRNA is high.
- If the concentration of the transcription factor is low, then the expression level of the mRNA is low.

Discrete logic models are suitable to implement this type of knowledge. Logic functions specify new states depending on current states.

Both types of knowledge can be directly implemented in PNFL models using a semi-continuous or a semi-discrete modeling approach. The semi-continuous approach is similar to the approach used in ODE models. Here, fuzzy logic systems are designed such that the resulting defuzzified output values represent marking changes. Per definition, functions that are assigned to arcs compute marking changes for incident places. Thus, the semi-continuous approach can be straightforwardly realized by directly assigning fuzzy logic systems to arcs or by assigning functions that totalize the defuzzified output value of several fuzzy logic systems to arcs. Multiple effects can be integrated as described above. Especially, a place can be influenced by multiple arcs. Fast or slow processes can be straightforwardly modeled by selecting consequent fuzzy sets with large or small centers of gravity.

The semi-discrete approach is similar to the approach used in Boolean or discrete multi-valued logic models. Here, fuzzy logic systems are designed such that the resulting defuzzified output values represent new markings. All fuzzy logic systems that propose new markings for the same place have to be integrated as described above and have to be embedded in a single function that additionally removes the old marking (**Figure 4.4**). Modeling of fast or slow processes using the semi-discrete approach can be done either by defining multiple consequent fuzzy sets and



$$f(A) = -M_k(p) + fls(A) \quad \Rightarrow \quad M_{k+1}(P) = M_k(P) + f(A)$$
$$\Leftrightarrow \quad M_{k+1}(P) = M_k(P) - M_k(P) + fls(A)$$
$$\Leftrightarrow \quad M_{k+1}(P) = fls(A)$$

Figure 4.4 Semi-discrete modeling. Fuzzy logic systems can be designed such that they propose new markings for entities, given the current markings of effectors. This modeling approach is similar to Boolean or discrete multi-valued logic models. If semi-discrete modeling is applied, fuzzy logic systems have to be embedded in other functions, such that the old markings of places are removed when transitions fire.

incorporating the current marking as antecedent into the fuzzy logic system, or by calculating a weighted mean between current marking and new marking (**Figure 4.5**). Examples for semi-continuous and semi-discrete models are given in the following Section 4.2.

4.2 PNFL Models Can Mimic Common Network Motifs

This section demonstrates the application of PNLF models to several well studied biological network motifs: feed-forward loop, negative feedback oscillator, positive feedback toggle switch and positive feedback one-way switch. Hereby, it is shown that PNFL models are suited to mimic the functionality of these motifs. The influence of antecedent fuzzy set design to the qualitative and quantitative behavior of models is discussed, and it is demonstrated that PNFL models can be analyzed analytically, by bifurcation diagrams, and phase planes.

4.2.1 Feed-Forward Loop

A feed-forward loop is composed of three genes: A transcription factor F regulates a co-factor C, which both jointly regulate a target gene T. Thus, the feed-forward loop has three regulatory interactions (F to C, F to T, C to T) which can be either activating or inhibiting, allowing for eight different combinations. The effects of F and C are integrated at the cis-regulatory elements of T. Two simple types of integration are AND-like or OR-like functions. Thus, 16 different configurations of regulatory effects and integrations are possible. Mangan and Alon [110] performed a theoretical analysis of these 16 types of feed-forward loops. They found that feed-forward loops can accelerate or delay a signal transduction, and serve as persistence detectors or pulsers, depending on the configuration of the regulatory interactions and integration. The most abundant type of feed-forward loop found in *E. coli* as well as in *S. cerevisiae* has three activating regulatory interactions and an AND-like integration [110]. This feed-forward loop causes a delay in initial signal transduction. Target gene T is not expressed immediately after its direct regulator F is inserted into the system, but only after co-factor C has reached a minimum expression level. On the other hand, expression level of T drops without delay as soon as F is removed. Due to the initial delay, the feed-forward loop can act as a persistence detector. A short pulse of F is not

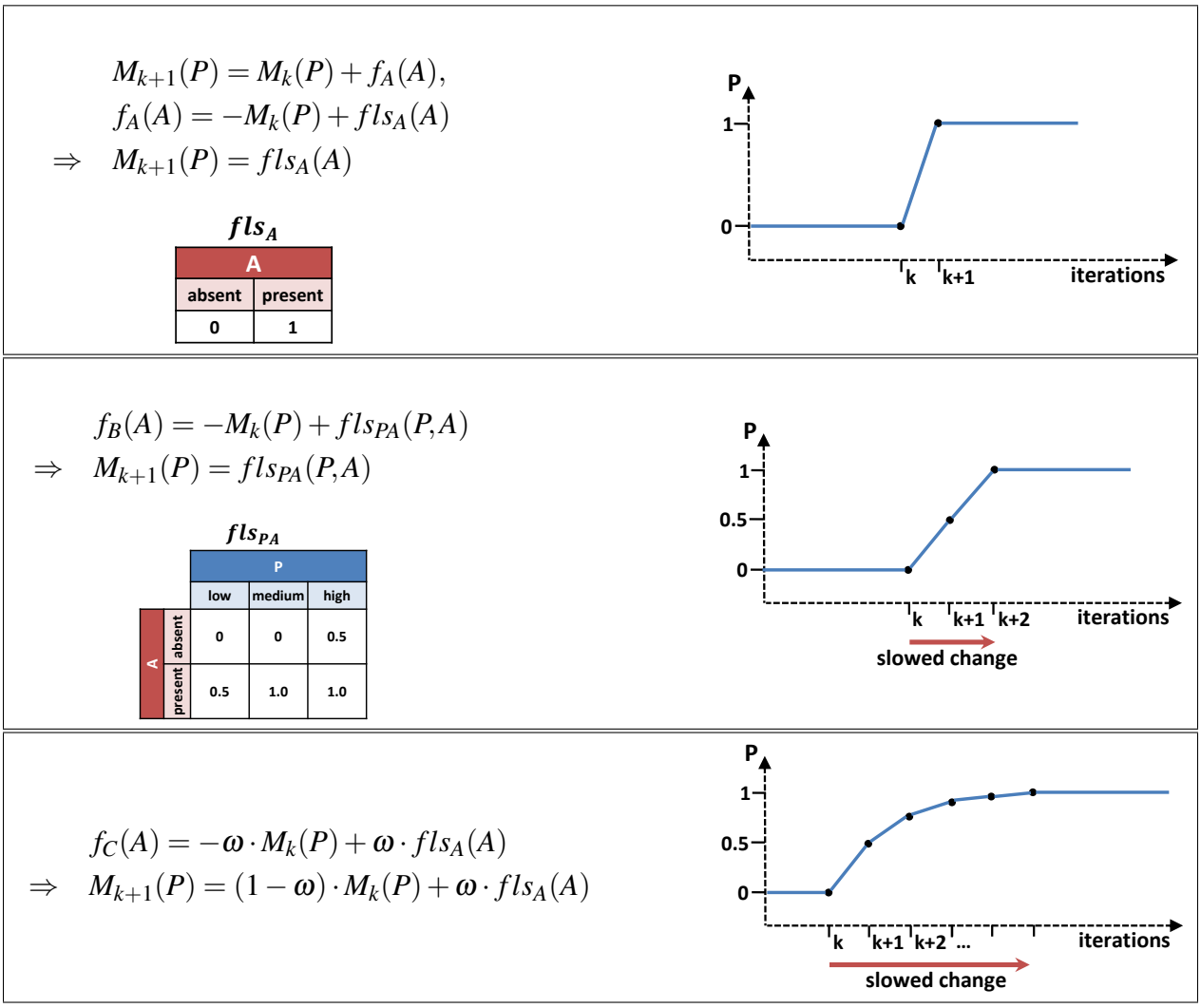


Figure 4.5 Modeling of slow state changes. If the semi-discrete modeling approach is applied, fuzzy logic systems are designed to propose target markings for places. Repeated firing of transitions must result in a convergence to the proposed target markings. If the functions that are assigned to arcs are designed such that they fully replace the current marking, target markings are reached after a single iteration (top). Here, fuzzy logic system fls_A proposes a new marking for place P depending on the state of A. If the state of A switches from *absent* to *present* at iteration k, then function f_A causes that the marking of P switches from 0 to 1 in one iteration. To slow down the marking changes, one could enforce intermediate steps by taking the current marking of the target entity in consideration (center). Fuzzy logic system fls_{PA} depends on the current states of P and A. If A is *present*, the state of P is increased by one step, from *low* to *medium*, or *medium* to *high*. Hereby, the centers of gravity of *low*, *medium*, and *high* are 0, 0.5, and 1. If the state of A switches from *absent* to *present* at iteration k, then function f_B causes the marking of P to switch from 0 to 1 in two iterations. As any further slowdown by one iteration can only be achieved by adding an antecedent fuzzy set and adding several rules, this approach is extremely inefficient. A more efficient approach is to design functions such that they do not fully replace the old marking, but instead calculate a weighted mean between old marking and the marking proposed by the fuzzy logic system (bottom). Function f_C calculates a weighted average based on an update-factor $\omega \in (0,1]$.

sufficient to cause a strong expression of T, while it is sufficient to cause a considerable response in expression of C.

We created a PNFL model of this type of feed-forward loop using the semi-discrete modeling approach (**Figure 4.6**). Thus, fuzzy logic systems assigned to arcs compute new states for target places. The current states of targets are used as inputs to fuzzy logic systems. Rules were defined such that only one-step changes of the target entities state are allowed, e.g. a state change from *low* to *medium* is allowed, but not from *low* to *high*. This slows down the state changes. Fuzzy logic system $fls_C(F,C)$ represents a simple activating function, that increases expression of C only if F is *present* and decreases it otherwise:

$$R_1 : IF\ F\ is\ present\ AND\ C\ is\ low\ THEN\ y_1\ is\ 0.5$$
$$R_2 : IF\ F\ is\ present\ AND\ C\ is\ medium\ THEN\ y_2\ is\ 1.0$$
$$R_3 : IF\ F\ is\ present\ AND\ C\ is\ high\ THEN\ y_3\ is\ 1.0$$
$$R_4 : IF\ F\ is\ absent\ AND\ C\ is\ low\ THEN\ y_4\ is\ 0.0$$
$$R_5 : IF\ F\ is\ absent\ AND\ C\ is\ medium\ THEN\ y_5\ is\ 0.0$$
$$R_6 : IF\ F\ is\ absent\ AND\ C\ is\ high\ THEN\ y_6\ is\ 0.5$$

The consequent fuzzy sets are identical to the antecedent fuzzy set *low*, *medium*, and *high* and are represented by their centers of gravity in the aforementioned rules.
Three experimental setting were simulated:

1. **Prolonged pulse.** Expression level of F is set to 1 and subject to exponential decay with rate 0.005. Expression levels of C and T increase to the maximum level and persist. The initial expression of T is delayed as compared to C.

2. **Long pulse.** Expression level of F is set to 1 and subject to exponential decay with rate 0.05. Expression levels of C and T increase to the maximum level. When F drops below 70 % of its maximum, both C and T expression levels start to decrease and drop to zero.

3. **Short pulse.** Expression level of F is set to 1 and subject to exponential decay with rate 0.5. Expression of C increases to 50 % of its maximum and drops to zero immediately after expression of F drops below 50 %. Expression of T increases only slightly to about 12 % before dropping to zero again.

The simulations show that the PNFL model of the feed-forward loop with three activating regulatory interactions and AND-like integration behaves qualitatively similar to the theoretical results of Mangan and Alon [110]. I.e. we observe a delay of the target gene's initial expression and observe that short signal pulses are filtered and do not cause a strong reaction of the target gene.

Fuzzy set definitions can be modified to change the quantitative behavior of the system. For example, fuzzy sets *absent* and *present* can be modified such that given the same signal, the response of the system is prolonged or shortened, or such that C and P decrease faster or slower. We will discuss this in more detail in the following. Fuzzy sets *absent* and *present* are realized as unbounded trapezoidal fuzzy sets and are defined by points $L = 0.3$ (right top of *absent* and left border of *present*) and $R = 0.7$ (right border of *absent* and left top of *present*). A signal

concentration smaller than L is considered as insufficient to trigger a response, a signal larger than R triggers the maximal response, and a signal between L and R triggers an intermediate response. We consider two special iterations in the time course of P. I_1 is the first iteration where P drops below the maximal value that was attained after the signal triggered, and I_2 is the first iteration where P drops to zero again. We consider four modified definitions of *absent* and *present*:

1. The distance between L and R is shortened. This causes I_1 to be shifted to the right and I_2 to be shifted to the left, The result is a prolonged maximal response to the signal, but the response drops more rapidly when the signal fades.
2. The distance between L and R is widened. This causes I_1 to be shifted to the left and I_2 to be shifted to the right. The systems response starts to drop early but reaches zero later.
3. L and R are shifted to the left. I_1 and I_2 are shifted to the right. The system already reacts to small signal concentrations.
4. L and R are shifted to the right. I_1 and I_2 are shifted to the left. The system reacts to high signal concentrations only.

None of the discussed modifications to fuzzy sets *absent* and *present* influences the overall qualitative behavior, i.e. the delay of signal transduction and the filtering of signal bursts, although the maximal length of filtered bursts may vary.

4.2.2 Negative Feedback Oscillator

Oscillating systems play an important role in cellular contexts, such as the circadian rhythm, the cell-cycle, or the formation of somites in embryos [111]. One of the most simple oscillating systems is a two-component negative feedback oscillator. An example would be a protein which inhibits the transcription of its own gene (**Figure 4.7A**). Thus, such a system consists of two components: a protein P and its mRNA R. Two processes occur here: translation of the protein which can only occur if the mRNA is present, and transcription of the mRNA which only occurs if the protein is absent (negative feedback). Such a system has to fulfill four requirements to allow oscillations: a negative feedback that allows that the states of affected entities return towards their initial states during oscillations; nonlinear kinetic laws that allow to over- or undershoot steady states, a sufficient time delay of the feedback signal such that the system can not settle on the steady state, and a similar timescale of reactions [111].

We created a PNFL model of this type of negative feedback oscillator (**Figure 4.7**) using the semi-qualitative modeling approach. To slow down state changes, the weighted averages of current states and outputs of fuzzy logic systems are computed and used as new states. The update-factor ω is 0.3. The compact functional notations of fuzzy logic systems fls_R and fls_P are:

$$fls_R(P) = \begin{cases} 1 & \text{if } P < 1 \\ \frac{1.5-P}{0.5} & \text{if } 1 \leq P \leq 1.5 \\ 0 & \text{if } 1.5 < P \end{cases}$$

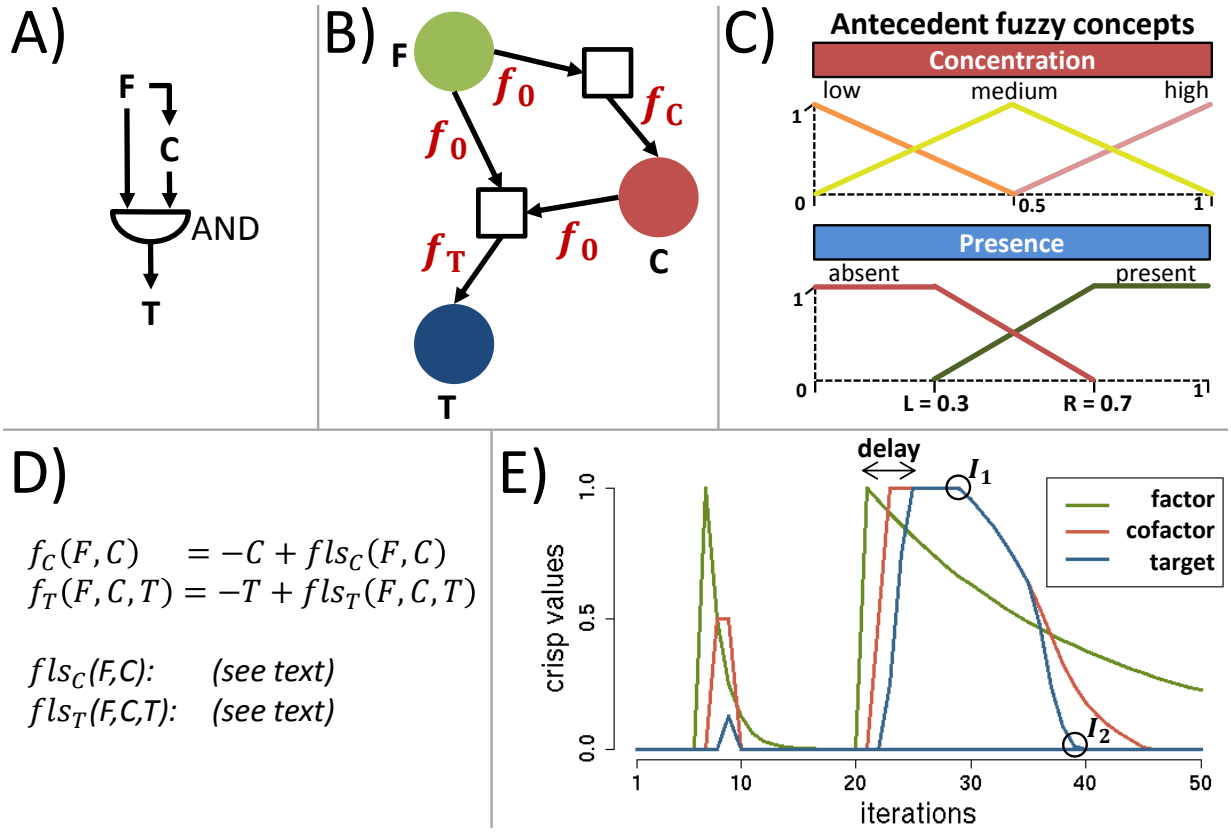


Figure 4.6 Feed-forward loop. This PNFL model represents a feed-forward loop (panel A). The expression levels of genes F, C, and T are represented by individual places. The integrated activating effects of F to T and C to T are represented by a single transition, as is the activation of C by F (panel B). We utilized the semi-discrete modeling approach. Thus, the two fuzzy logic systems assigned to the output arcs of the transitions compute new states for output places (panel D). These states could be either *low* ($\bar{y}_{low} = 0$), *medium* ($\bar{y}_{medium} = 0.5$), or *high* ($\bar{y}_{high} = 1$, panel C). Fuzzy logic systems use the current expression level of the target entity as input and fuzzify it using the three fuzzy sets *low*, *medium*, and *high* mentioned before. The other antecedents were fuzzified using fuzzy sets *absent* and *present* (panel C), i.e. it is assumed that F and C must reach a certain expression level before they affect their target genes. Rules are designed such that they allow only one-step changes of target expression, e.g. if the current expression is *low*, the consequent expression may either be *low* or *medium*, but not *high*. This slows down expression changes. Fuzzy logic system $fls_C(F,C)$ represents a simple activating function, increasing expression of C only if F is *present* and decreasing it otherwise (see main text for rules). Fuzzy logic system $fls_T(F,C,T)$ is an AND-like integration of two activating functions (Section 3.2.3) which increases expression of T only if F and C are *present* and decreases it otherwise. The according rule system has $2 \cdot 2 \cdot 3 = 12$ rules. It is omitted for brevity. The time course of expression level of F is not influenced by the PNFL model, but provided according to the experimental setting. Panel E shows the time courses of two simulated experiments: a short signal burst (before iteration 10) and a long signal burst (after iteration 20). See the main text for a description.

$$fls_P(R) = \begin{cases} 0 & \text{if } R < 0.4 \\ 2 \cdot \frac{R-0.4}{0.2} & \text{if } 0.4 \leq R \leq 0.6 \\ 2 \cdot \frac{0.8-R}{0.2} + 4 \cdot \frac{R-0.8}{0.2} & \text{if } 0.6 < R \leq 0.8 \\ 4 & \text{if } 0.8 < x \end{cases}$$

The two processes of the system, the (activating) translation and the (inhibiting) transcription reaction form the negative feedback. The definitions of antecedent fuzzy concepts and fuzzy logic systems result in (coarse-grained) piecewise-linear functions with a nonlinear response to input values (Section 3.2.1). The time delay is realized by using R as an intermediate entity instead of modeling a direct negative effect of P to itself. The half-life of both entities R and P is three iterations, so the time scale of reactions is similar. The PNFL models exhibits sustained oscillations for all combinations of initial concentrations and expression levels.

Proposition There is no stable steady state, but only an unstable focus for $\omega \in (0,1)$. This can be proven by contradiction.

Proof Independent of the update-factor ω, a steady state is reached if at any iteration i the following holds:

$$M_k(P) = fls_P(M_k(R)) \quad \wedge \quad M_k(R) = fls_R(M_k(P))$$

For brevity, we define $P = M_k(P)$ and $R = M_k(R)$ for the following proof. We consider six cases based on the possible crisp values of P and R.

case 1: Assume $(P < 1) \quad \Rightarrow \quad R = fls_R(P) = 1 \quad \Rightarrow \quad P = fls_P(R) = 4$ (contradiction)

case 2: Assume $(1 \leq P \leq 1.5) \wedge (R < 0.4) \quad \Rightarrow \quad P = fls_P(R) = 0$ (contradiction)

case 3: Assume $(1 \leq P \leq 1.5) \wedge (0.4 \leq R \leq 0.6)$

$$\Rightarrow \quad P = fls_P(R) = 2 \cdot \frac{R-0.4}{0.2} \quad \Rightarrow \quad R = fls_R(fls_P(R)) = \frac{1.5 - 2 \cdot \frac{R-0.4}{0.2}}{0.5}$$
$$\Rightarrow \quad R \approx 0.5238 \text{ and } P \approx 1.238$$

case 4: Assume $(1 \leq P \leq 1.5) \wedge (0.6 < R \leq 0.8)$

$$\Rightarrow \quad R = fls_R(P) = \frac{1.5 - P}{0.5}$$
$$\Rightarrow \quad P = fls_P(fls_R(P)) = 2 \cdot \frac{0.8 - fls_R(P)}{0.2} + 4 \cdot \frac{fls_R(P) - 0.8}{0.2}$$
$$\Rightarrow \quad P \approx 1.0476 \text{ and } R \approx 0.905 \text{ (contradiction)}$$

case 5: Assume $(1 \leq P \leq 1.5) \wedge (0.8 < R) \quad \Rightarrow \quad P = fls_P(R) = 4$ (contradiction).

case 6: Assume $(1.5 < P) \quad \Rightarrow \quad R = fls_R(P) = 0 \quad \Rightarrow \quad P = fls_P(R) = 0$ (contradiction).

Thus, there is only one fixed point at $R \approx 0.5238$ and $P \approx 1.238$ (case 3). The Jacobian matrix for case 3 is

$$J = \begin{pmatrix} \frac{\partial f_R(R,P)}{\partial R} & \frac{\partial f_R(R,P)}{\partial P} \\ \frac{\partial f_P(R.P)}{\partial R} & \frac{\partial f_P(R,P)}{\partial P} \end{pmatrix} = \begin{pmatrix} 1-\omega & -2\cdot\omega \\ 10\cdot\omega & 1-\omega \end{pmatrix}$$

For any $\omega \in (0,1)$ both eigenvalues of the Jacobian are complex with positive real parts equal to $(1-\omega)$. Thus, the fixed point is an unstable focus, **q.e.d.**

4.2.3 Positive Feedback Toggle Switch

Positive feedback may create a discontinuous switch, i.e. the cellular response to a signal S changes abruptly if the value of S crosses a critical value [112]. For example, the concentration of a protein A in the system is *low* until the signal S crosses the critical value. Then the concentration switches abruptly to *high*. So, the system resembles some kind of on-off-switch, and switching is controlled by the signal S. If switching is reversible, the system can be denominated as *toggle switch*, i.e. if decreasing S until it crosses a possibly different critical value causes the cellular response to abruptly change back to the initial state. Some examples for toggle switches are the *lac* operon in bacteria or the activation of M-phase-promoting factor in frog egg extracts [112].

We created a PNFL model of a positive feedback toggle switch (**Figure 4.8**) using the semi-continuous modeling approach. Multiple effectors are integrated using multidimensional rule tables. Most fuzzy logic systems use triangular antecedent fuzzy sets *low* and *high*. The exception is the antecedent fuzzy concept used for fuzzifying of A in the decay process of B. Here, we designed a narrow transition from trapezoidal fuzzy set *absent* to *present* between concentrations 0.2 and 0.4 The synthesis of A is defined by the following rules:

$$R_1 : IF\ S\ is\ low\ AND\ B\ is\ high\ THEN\ y_1\ is\ 0.0$$
$$R_2 : IF\ S\ is\ low\ AND\ B\ is\ low\ THEN\ y_2\ is\ 0.0$$
$$R_3 : IF\ S\ is\ high\ AND\ B\ is\ high\ THEN\ y_3\ is\ 0.05$$
$$R_4 : IF\ S\ is\ high\ AND\ B\ is\ low\ THEN\ y_4\ is\ 0.2$$

Rules R_1 and R_2 define the minimal synthesis rate of A in case the signal S is *low*, effectively independent of the state of inhibitor B. Rule R_4 defines the maximal synthesis rate of A in case S is *high* and B is *low*. Rule R_3 states that an increase in signal S increases A also if the inhibitor is *high*, although at a reduced rate. It is crucial for the toggle switch property of the system that it can always react to state changes of S, as otherwise it would exhibit an irreversible one-way switch behavior.

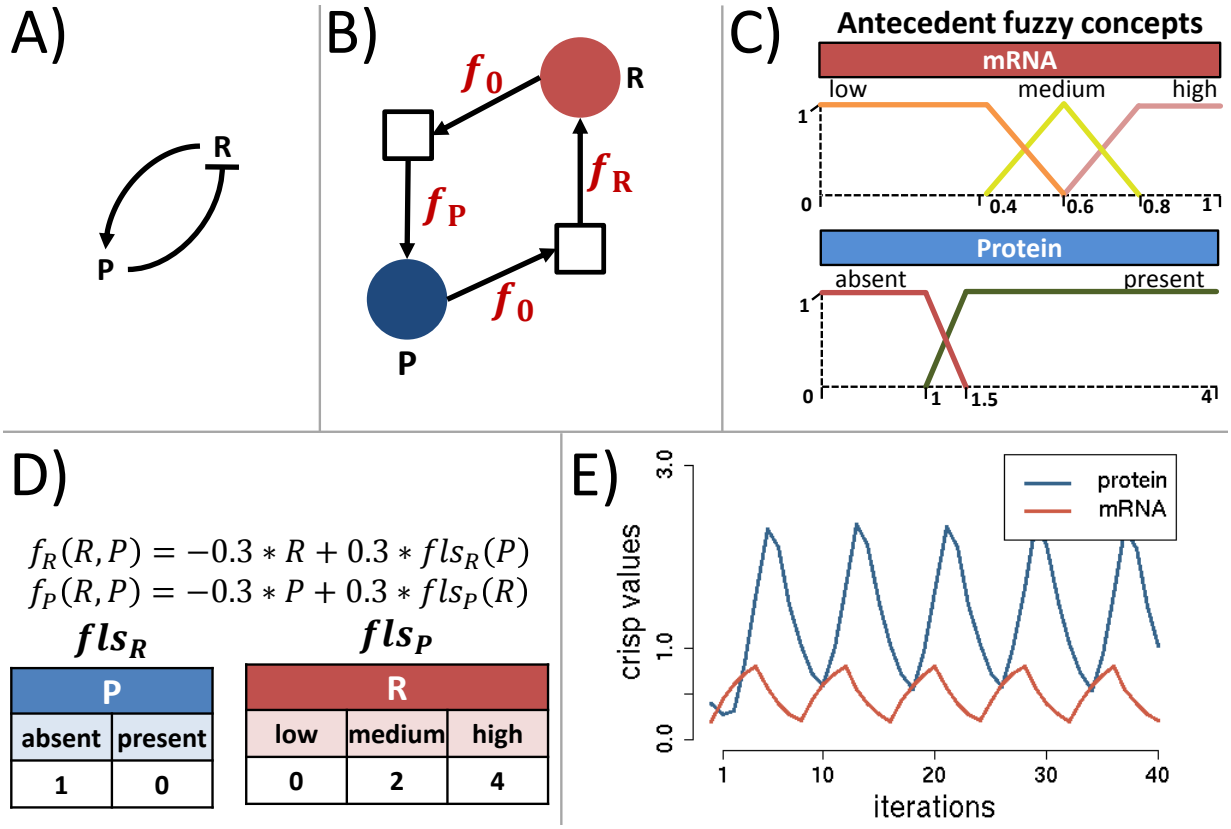


Figure 4.7 Negative feedback oscillator. This PNFL model represents a negative feedback oscillator (panel A). Two places P and R represent protein concentration and mRNA expression level, two transitions represent transcription and translation reactions (panel B). Protein concentration is fuzzified using trapezoidal fuzzy sets *absent* and *present*, while mRNA expression levels are fuzzified using fuzzy sets *low*, *medium*, and *high* (panel C). The semi-discrete modeling approach is utilized. To slow down state changes, the weighted averages of current states and outputs of fuzzy logic systems are computed and used as new states (panel D). Fuzzy logic systems fls_R and fls_P compute the new states of R and P, respectively. They mimic simple inhibiting and activating functions. The PNFL model exhibits oscillating behavior (panel E).

The bifurcation diagram in **Figure 4.8E** shows steady state levels of A and B depending on the concentration of S. If the concentration of S is very low the system is in a steady state with low A and high B. If S is increased, A increases slowly as well while B stays at its maximum (symbolized by black arrows in **Figure 4.8E**). If A reaches a critical concentration (point I_1 in the time course), an abrupt change of concentrations A and B occurs. The system switches its state to high A and low B. This system state can be reversed by decreasing S (symbolized by gray arrows). A decreases proportional to S and eventually reaches another critical concentration (point I_2 in the time course). Here, the system switches back to its initial state with low A and high B. If S is between I_1 and I_2, two steady states of A and B are possible. Which of those is attained depends on past states of the system.

The design of fuzzy sets *absent* and *present* influences the quantitative behavior of the system. Shifting the borders of the fuzzy sets causes a shift of critical points I_1 and I_2. If the borders are shifted to higher concentrations, higher levels of A (and indirectly of S) are necessary before the state-switch occurs, and *vice versa*. Widening the transition zone of fuzzy sets *absent* and *present* increases the distance between I_1 and I_2, while narrowing the transition zone decreases the distance. The qualitative behavior of the system is not influenced by such design changes, it is still a toggle switch.

4.2.4 Positive Feedback One-Way Switch

Another type of switch-like systems are positive feedback one-way switches. Hereby, the cellular response to a signal is irreversible, i.e. if the signal Z crosses a critical value, the concentration of a protein X changes abruptly from *low* to *high* but will not change back if Z is decreased. Such one-way switches play an important role in developmental processes, in general by controlling and determining cell fates [113].

We created a PNFL model of an one-way switch (**Figure 4.9**) using the semi-continuous modeling approach. Multiple effectors are integrated using generalized means which resemble AND- or OR-like functions.

If the signal Z is absent, the system has two stable states, either X is *low* and Y is *high* (state $low/high$) or *vice versa* (state $high/low$). Inserting the signal Z to the system causes an abrupt switch from state $low/high$ to state $high/low$ (**Figure 4.9E**), as the system now has only one stable steady state which is reached independent of the starting values of X and Y. State $high/low$ can not be reversed by decreasing Z. Thus, this state is irreversible. The potential behavior of the system can be visually assessed by representing potential state changes by phase planes (**Figure 4.10**).

4.3 Discussion

PNFL models join a Petri net based graphical representation of the structure with a fuzzy logic based representation of states and processes of a system. Hereby, different aspects or properties of entities that are of relevance to a model are represented by places. The current state of an entity with respect to a property, e.g. its concentration, is stored as a real-valued crisp number. The

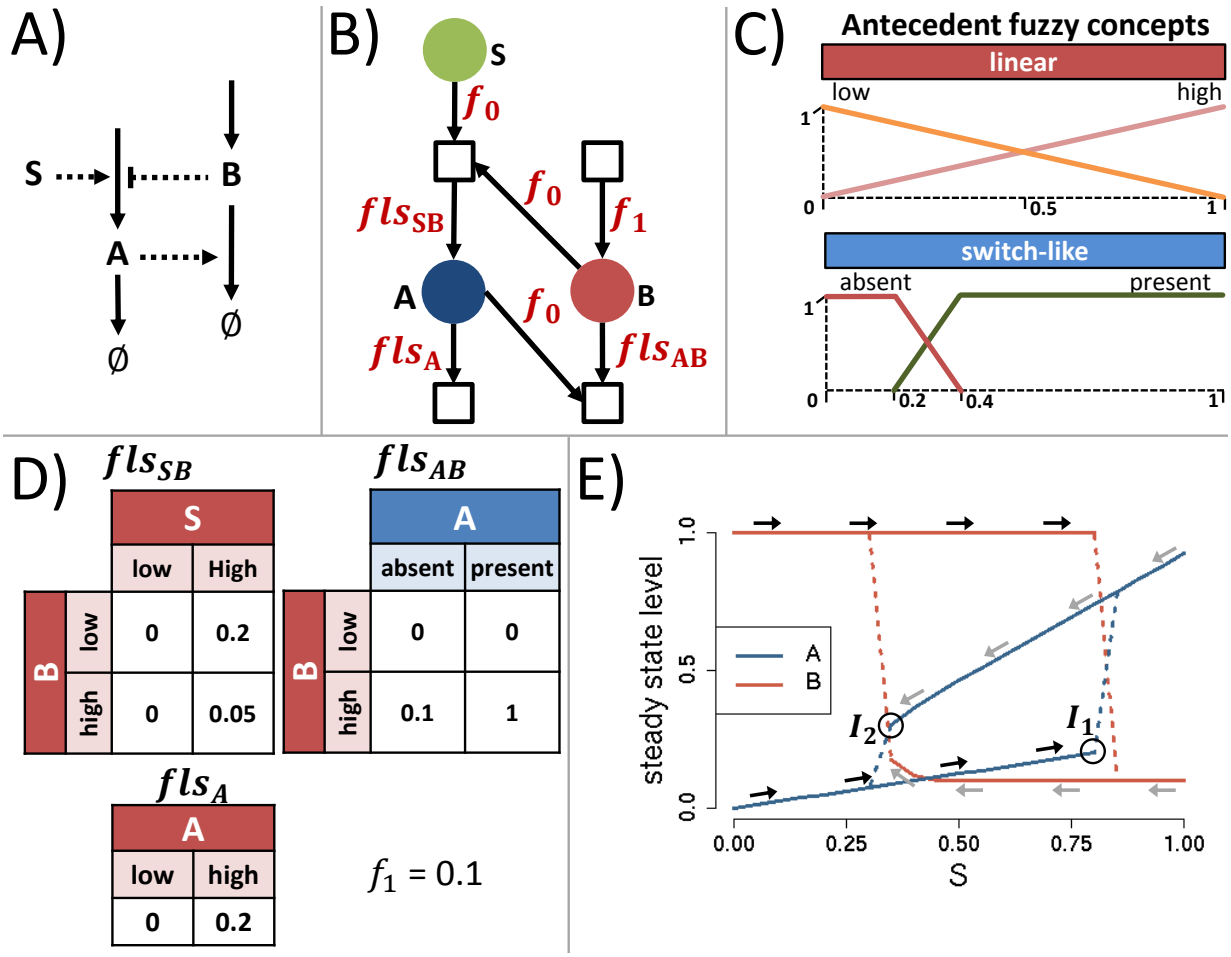


Figure 4.8 Positive feedback toggle switch. This PNFL model represents a positive feedback toggle switch (panel A). The signal S activates the synthesis of Protein A, which enhances the decay of a protein B. Protein B inhibits the synthesis of A. Thus, activation of A leads to the suppression of its inhibitor (positive feedback). Protein A is decayed independently and protein B is synthesized at a constant rate. Signal and protein concentrations are represented as places, the two synthesis and decay processes are represented as transitions (panel B). The concentration of S is not influenced by the PNFL model, but provided according to the experimental setting. The definitions of antecedent fuzzy sets and fuzzy logic systems are very simple (panel C). We utilize the semi-continuous modeling approach, i.e. fuzzy logic systems compute changes of the current states. Multiple effectors are integrated using multidimensional rule tables. Most fuzzy logic systems use triangular antecedent fuzzy sets *low* and *high*, resulting in a linear response to crisp value changes. The exception is the antecedent fuzzy concept used for fuzzifying of A in the decay process of B (fls_{AB}, panel D). Here, a narrow transition from trapezoidal fuzzy sets *absent* to *present* was implemented. This causes the abrupt response of the system to changes of A's concentration. The bifurcation diagram shows steady state levels of A and B depending on the concentration of S (panel E). If the concentration of S is in the intermediate range between points $crit_1$ and $crit_2$, two steady state levels of A and B are possible (low and high). Which state level is achieved depends on how S was changed, i.e. whether its concentration was low and then increased or *vice versa*.

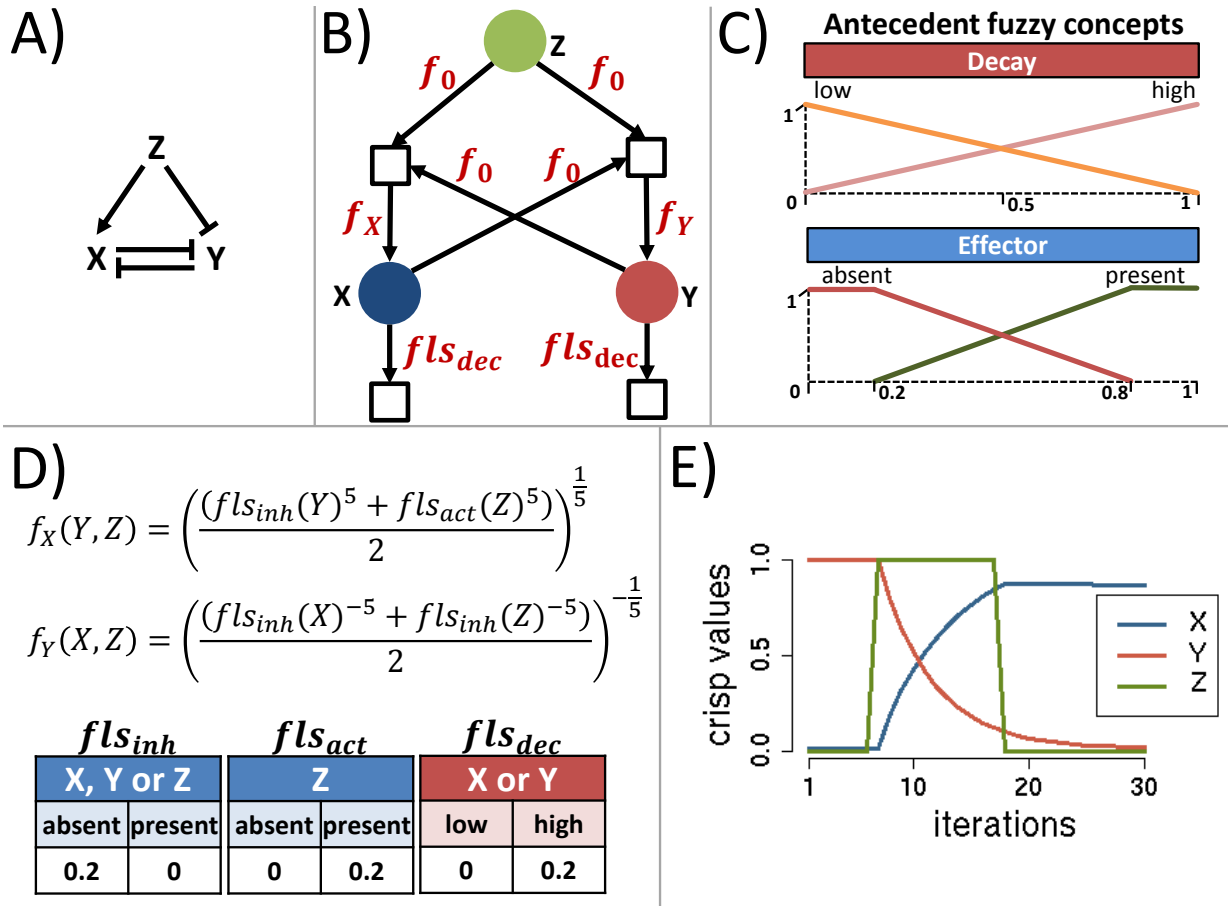


Figure 4.9 Positive feedback one-way switch. This PNFL model represents a positive feedback one-way switch (panel A). It comprises a signal Z which activates a transcription factor X and inhibits the transcription factor Y. X and Y are mutually inhibiting, thus they inhibit their own repressor (positive feedback). We implemented two synthesis and two decay processes using semi-continuous modeling (panel B). Simple triangular and trapezoidal fuzzy sets are used as antecedents (panel C). The synthesis of transcription factor X or Y is influenced by Z and by the respective other transcription factor. Thus, multiple entities affect a single synthesis process. The effect of each entity (Z and a transcription factor) is represented by an individual fuzzy logic system with one antecedent. The defuzzified output values are integrated using generalized means which resemble AND- or OR-like functions (panel D). The concentration of Z is not influenced by the PNFL model, but provided accordingly to the experimental setting. If Z is inserted into the system, it switches from one steady state (X is *low* and Y is *high*) to the other steady state (X is *high* and Y is *low*). This state switch can not be reversed by decreasing Z (panel E).

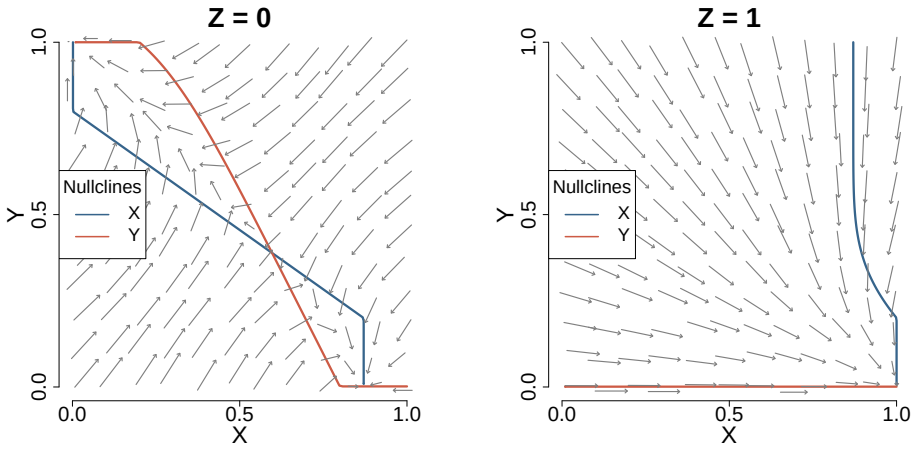


Figure 4.10 **Phase planes** can be used to assess the behavior of the one-way switch. Arrows indicate state changes of X and Y given a fixed Z. Each arrow points from $(M_k(X), M_k(Y))$ in direction of $(M_{k+1}(X), M_{k+1}(Y))$. Given an absent signal Z, the system has two stable steady states at X/Y is $low/high$ and X/Y is $high/low$ (left). Depending on the initial states of X and Y, one of these steady states is reached. If the signal Z is set to one, the phase plane changes and only one stable steady state remains at X/Y is $high/low$ (right). Independent of the current states of X and Y, the system converges to this steady state. If after convergence Z is again decreased, the system will stay in state $high/low$. Thus, the state change can not be reversed by the signal.

transitions represent processes and are connected to effectors and targets by arcs. The effects of a process to its targets are calculated using fuzzy logic systems, thus they define the functionality of the model. The fuzzy set based representations are realized as part of the fuzzy logic systems. Hereby, several different fuzzy set based descriptions can be applied to describe the same aspect/property of an entity, depending on the functional implications to the different processes. This allows for a high degree of flexibility with respect to the functionality of a system. Furthermore, the real-valued markings can be easily plotted or compared to experimental data. PNFL models have an inherent graphical representation of entities and interactions due to their Petri net structure. This facilitates understanding of the effector-target relationships between entities and allows for structural analysis of networks, e.g. the degree distribution, clustering coefficients, or causal relationships.

Although the complexity of fuzzy logic systems is restricted by possible combinations of antecedent and consequent fuzzy sets, fuzzy logic systems can still be very flexible and allow for complex influences of effector to target entities, as fuzzy sets can be designed freely. As the total number of possible combinations can be huge if an entity is influenced by many effectors, the number of rules that have to be necessarily defined is huge as well. Due to the possibility to totalize individual effects using generalized mean functions, PNFL models allow for a significant simplification of rules that is suited to represent simple AND- or OR-like relations between effectors.

Section 4.2 demonstrates that PNFL models can be used to successfully model small biological systems, namely several well studied network motifs. The PNFL models are able to capture

complex behavior like oscillations, or toggle and one-way switches. Standard analysis methods can be applied to investigate the theoretical behavior of PNFL models, e.g. phase planes to visually assess convergence to steady states, or bifurcation diagrams to show systems behavior in dependence to changes of an entities state. Thus, PNFL models can be conveniently analyzed using well established techniques.

A comparison to ODE and discrete logic models Semi-quantitative and semi-qualitative modeling approaches for PNFL models were introduced, which can be freely chosen according to design needs (Section 4.1.3). These approaches resemble the functional representations used in ODE and discrete logic models, respectively (Section 1.2). The ODE model-like semi-quantitative modeling is especially suited if several independent processes that affect the same entities should be implemented, for example, if decay and production processes are modeled explicitly. Here, individual functions are used to calculate state changes that can be simply summed to derive their total effect.

The discrete logic model-like semi-qualitative modeling is based on functions that propose new states for entities. If several processes affect an entity simultaneously, then their individual effects have to be totalized to derive a joint new state. For example, decay and production are realized by a single function, which is used to calculate the steady state that results from the equilibrium of production and decay.

In the following Section 5, PNFL is used to model a bacterial transcription/translation system. The modeling results are compared to an ODE model of the same system. Further, it is shown that the qualitative PNFL models are suitable for testing several hypotheses about the systems functionality.

Author's contribution The author developed the Petri net and fuzzy logic modeling technique and applied it to model various biological motifs.

Chapter 5

A Model of a Cell-Free Transcription/Translation System

The following section provides an application example of PNFL modeling of a simple biological system: green fluorescent protein (GFP) expression in a cell-free *in vitro* transcription/translation system (**Figure 5.1A**). We introduce and compare an ODE and a PNFL model of this system and illustrate that the qualitative PNFL model is suitable for model discrimination. The work presented here was performed during a joint project with Tobias Stögbauer, and parts of it have been published in [104]. The research focus of the project was to study the gene expression kinetics of a cell-free gene expression system using a predictive computational model. The author developed, implemented, and evaluated the computational models, while TS performed all experiments and participated in ODE model creation.

The investigated biological system is a bacterial (*E. coli* like) transcription and translation system. All processes occur *in vitro* in a cell-free environment, i.e. the system was reconstituted from individual components necessary for transcription and translation reactions only (polymerases, ribosomes, tRNAs, NTPs, etc). All other cellular components were omitted. The template DNA is provided by the experimenter; in this case plasmids encoding the GFP protein. In short, the three main processes of this system are:

- transcription of GFP-mRNA from template DNA by RNA-polymerases and other necessary transcription components,
- translation of pre-mature GFP from this mRNA by ribosomes and other necessary translation components,
- and finally unmediated maturation of GFP. Mature GFP is the end-product of the system.

Several measurements of mRNA and GFP time courses have been performed under varying experimental settings (Section 5.1). The computational models of this system (Section 5.2) were reverse-engineered such that they either allow a concurrent fit to GFP and mRNA concentrations (ODE model) or give a qualitatively adequate description of GFP and mRNA kinetics (PNFL model). These models allow to test hypotheses about the processes occurring in the cell-free *in vitro* system. It was observed that transcription and translation slow down and cease after several

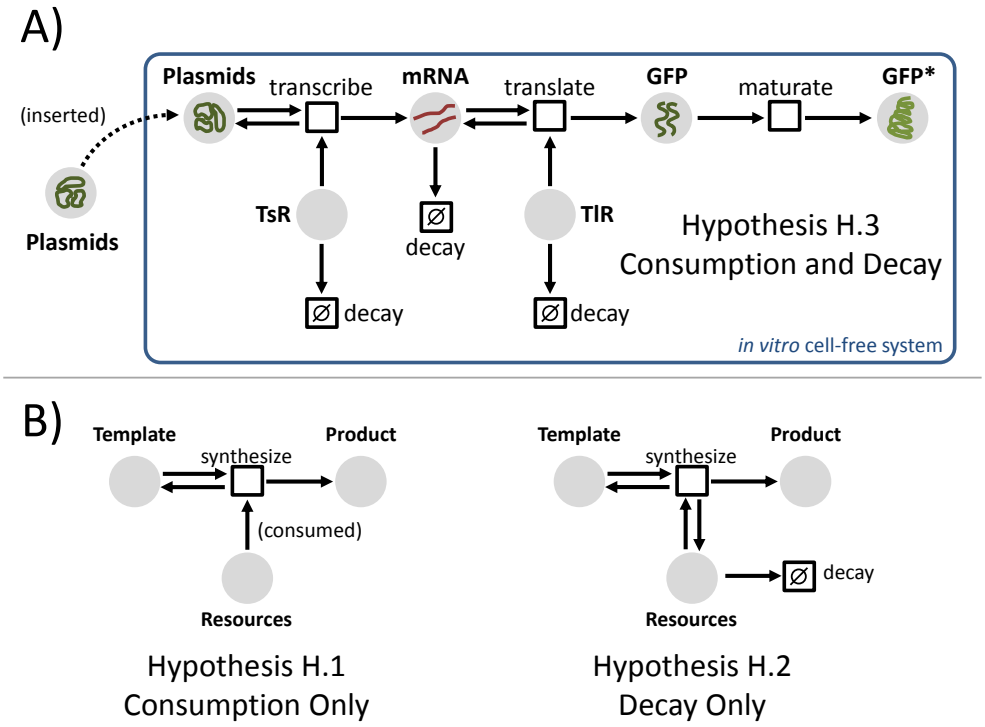


Figure 5.1 A) A sketch of the bacterial transcription/translation system. Template DNA (plasmids) is inserted into the *in vitro* environment. The plasmids encode a green fluorescent protein (GFP) gene. The GFP gene is transcribed and mRNA subsequently translated to premature GFP. The latter maturates unmediated. The concentration of mature GFP (GFP*) can be measured by a spectrometer. Transcription and translation are influenced by diverse molecular components. They are subsumed as transcription resources (TsR) and translation resources (TlR). The cell-free system is free of exonuclease and protease. Thus plasmids, GFP, and GFP* are not degraded. B) A sketch of the hypotheses about transcription/translation resource reduction. TsR and/or TlR are depleted several hours after activation of the cell-free system (Section 5.1). Whether they are consumed by the transcription and translation processes or whether they decay is not known. The computational models are used to test three hypotheses H.1, H.2, and H.3 explaining TsR/TlR reduction: consumption only (H.1, left), decay only (H.2, right), or both (H.3, shown in panel A).

hours. This is caused by the depletion of some molecular components necessary for transcription and translation (see Section 5.1 for a discussion). We propose three hypotheses that might explain the depletion:

Hypothesis H.1 Some necessary molecular components are consumed (or degraded) by the translation and transcription processes. Possible translation/transcription independent decay processes occur at very slow rates and their influence is negligible. Candidate molecules could be NTPs, tRNAs, etc.

Hypothesis H.2 Some necessary molecular components decay at significant rates while their

consumption is negligible. This might apply to components that can be reused, like polymerases or ribosomes.

Hypothesis H.3 A mixture of hypotheses H.1 and H.2, i.e. consumption and degradation have a considerable influence.

The computational models are capable to distinguish between these hypotheses using the available experimental data (Section 5.3). The true hypothesis allows for a better fit to the experimental data, which can be used for model selection using the ODE model. Each hypothesis actually implies different qualitative behaviors of the GFP time courses. This can be predicted using the PNFL model.

5.1 Experimental Methods and Acquired Data

We will only provide a brief description of the cell-free expression system and its constituents and focus on those aspects which are important to consider for model creation. All further experimental details concerning plasmid DNA creation, cell-free gene expression kit, data acquisition techniques, etc. can be found in [104].

To express green fluorescent protein (GFP) a reconstituted cell-free *in vitro* transcription/-translation kit was used (PURExpress kit, [114]). The cell-free system contains about 50 purified components necessary to allow transcription and translation as found in *E. coli*: most importantly T7 RNA-polymerases, ribosomes, tRNAs, and NTPs. A detailed and quantitative composition is not provided by the manufacturer, but all components are present at high copy numbers. Due to the controlled reconstitution of components the system is essentially exonuclease, RNAse, and protease free.

Due to the nature of the reconstituted cell-free expression system, one can make the following prior assumptions for model creation:

- No stochastic effects on reactions, as all constituents are present at high copy numbers.
- No replenishment of any constituents of the system. Thus, components necessary for transcription or translation can get exhausted.
- No plasmid degradation or decay. DNA is quite stable and the cell-free system is exonuclease free. Thus, the total concentration of template DNA can be assumed constant during all experiments.
- No active mRNA degradation, as the cell-free system is RNAse free. Thus, mRNA might decay due to its instability only, if at all.
- No GFP degradation or decay. GFP is quite stable and the cell-free system is protease free. Thus, one can assume that (maturated) GFP is not degraded and does not decay during experiments.

Acquired Data and discussion of the observed kinetics Six experimental settings and respective series of measurements were available for model fitting and validation (**Figure 5.2**).

Experiment 1: GFP expression kinetics for different concentrations of template DNA. Immediately after activation of the cell-free system by mixing of all constituents and heating, template DNA was added to the system. GFP concentrations were measured in regular intervals for about 5 h. Measurements were available for template DNA concentrations spanning five orders of magnitude: 340 femtomolar (fM), 3.4 picomolar (pM), 34 pM, 340 pM, and 3.4 nanomolar (nM).

Experiment 2: GFP expression kinetics after delayed addition of template DNA. 3.4 nM template DNA was added at six different time points after activation of the cell-free system by mixing of all constituents and heating: 0 min, 37 min, 73 min, 112 min, 153 min, and 187 min. GFP concentrations were measured in regular intervals from the addition time until about 5 h after activation of the cell-free system.

Experiment 3: GFP maturation kinetics. After the first 3 h of expression (protein levels were still rising), ribosomes were inactivated to stop translation and thus any further increase in premature GFP concentration. After inactivation of the ribosomes, GFP concentrations were measured in regular intervals for about 30 min.

Experiment 4: mRNA synthesis kinetics for different concentrations of template DNA. Immediately after activation of the cell-free system, template DNA was added to the system: 6.8 nM, 1.7 nM, and 340 pM. mRNA concentration was measured in regular intervals for about 8 h (6.8 and 1.7 nM template) or 4 h (340 pM template).

GFP concentration saturates at different levels depending on template DNA concentration (Experiment 1). The saturation time after about 4 h seems to be independent of template DNA concentration. A steady-state between synthesis and reduction of GFP can be ruled out as a reason for the saturation, since GFP is not degraded and does not decay (as discussed above). This is additionally supported by Experiment 3, where synthesis was blocked but still no GFP reduction was observed. Thus, the saturation of GFP has to be caused by a cessation of the translation process.

The GFP-mRNA as template for translation is still present when GFP concentration saturates (Experiment 4). In fact, mRNA concentrations continually increase for several hours after GFP saturation. Thus, some other components necessary for translation (e.g. ribosomes, tRNAs, NTPs) are either completely consumed and/or decayed and cause the cessation of the translation process. A consumption of translation components can not be the only reason for cessation, as then GFP concentrations would saturate at the same level independent of template DNA concentration. GFP concentration saturates at different levels depending on addition time (Experiment 2). This indicates that some components degrade after activation of the cell-free system, independent of the occurrence of any transcription/translation. Whether this transcription/translation independent decay is the only reason for the expiration of protein synthesis, or whether some partial depletion of (other) components has an additional influence, can not be easily deduced from the available data.

The concentration of mRNA saturates as well (Experiment 4). This occurs much later as compared to GFP, i.e. after 8 h or later. As mRNA is not as stable as DNA or GFP, it might decay at a noticeable rate. A steady-state between synthesis and decay might cause this saturation. Saturating mRNA concentration was observed for 6.8 nM template DNA only. mRNA levels for 1.7 nM and 340 pM templates were still rising at the end of the measurement interval. Thus, one can not decide whether they saturate on the same level, or not. It was observed that the saturation times are dependent on template DNA concentrations. Thus, a DNA-independent decay of transcription components can not be the only reason for expiration of mRNA synthesis.

Absolute GFP concentrations of Experiments 1 and 2 differ strongly. For example, based on the same amount of template DNA (3.4 nM at addition time 0 min) we achieved 300 nM GFP yield in Experiment 1 and about 1000 nM in Experiment 2. This is presumably caused due to different conversions of the arbitrary fluorescence units to molar, and results in a linear scaling of concentrations in Experiment 2 by a factor of 3.64 as compared to Experiment 1.

5.2 Computational Models of the Cell-Free System

We aim to devise a quantitative ODE model that fits the data sets presented so far (**Figure 5.2**), as well as a PNFL model that qualitatively reproduces observed kinetics. A guiding principle of model development was to keep the number of free parameters as small as possible while still explain aforementioned experimental observations:

1. DNA concentration dependent saturation levels (**Figure 5.2**, top).
2. DNA concentration independent cessation of translation but not transcription after about 4 h (**Figure 5.2**, center).
3. Slow DNA concentration dependent saturation of mRNA (**Figure 5.2**, bottom right).

The final ODE and PNFL models described in this section result from the model selection process described in Section 5.3.

5.2.1 The ODE Model

The core species of the model are DNA, mRNA, GFP and maturated GFP (GFP*). Initial DNA concentrations are known from the experimental settings, while the concentrations of other core species are initially set to zero. The sets of molecules influencing transcription and translation are not fully known, nor are the initial concentrations of individual molecule types. Thus, variables TsR and TlR were introduced which represent the pools of all transcription and translation resources (polymerases, ribosomes, tRNAs, NTPs, and other components). The derivation of meaningful initial values for variables TsR and TlR prior to optimization is not reasonable for the same reasons. Thus, initial values were set to 1 nM and scaling factors were introduced.

Several kinetics for transcription and translation were tested (including mass-action, Hill and Michaelis-Menten kinetics) to cover several possible types of processes: from simple concentration dependent reactions to cooperative and saturation effects. We found that the observed

saturation levels of mRNA and GFP are not adequately captured by mass-actions kinetics for transcription or translation, i.e. when using mass-action kinetics the protein yield for either low or high template DNA concentrations could be fitted, but not both (data not shown). Adopting Hill functions for these processes results in very good fits for sensible DNA concentrations and insertion times. However, optimized Hill exponents n (Section 3.2.1) were very close to one, thus Hill equations were effectively reduced to Michaelis-Menten equations. The final, best fitting model is given by the following set of differential equations:

$$\frac{d}{dt}[mRNA] = \frac{k_{ts} \cdot [TsR] \cdot [DNA]}{m_{ts} + [DNA]} - \delta_{mRNA} \cdot [mRNA] \tag{5.1}$$

$$\frac{d}{dt}[GFP] = \frac{k_{tl} \cdot [TlR] \cdot [mRNA]}{m_{tl} + [mRNA]} - k_{mat} \cdot [GFP] \tag{5.2}$$

$$\frac{d}{dt}[GFP^*] = k_{mat} \cdot [GFP] \tag{5.3}$$

$$\frac{d}{dt}[TsR] = -\frac{k_{TsR} \cdot [TsR] \cdot [DNA]}{m_{ts} + [DNA]} \tag{5.4}$$

$$\frac{d}{dt}[TlR] = -\frac{\delta_{TlR} \cdot [TlR]}{m_{TlR} + [TlR]} \tag{5.5}$$

k_{ts}, k_{tl}, and k_{mat} are the rates of transcription, translation and GFP maturation. m_{ts} and m_{tl} are Michaelis-Menten constants. δ_{mRNA} is the decay rate of $mRNA$. Degradation of DNA and GFP is neglected (see Section 5.1). The transcription resources TsR are consumed by the transcription process with rate k_{TsR}, and thus depend on template DNA concentration, whereas translation resources TlR decay independent of the presence of DNA and thus independent of a translation process with rate δ_{TlR} and Michaelis-Menten constant m_{TlR} (see Section 5.3). As the true concentrations of TsR and TlR are not known, rates k_{ts}, k_{tl}, k_{TsR} as well as δ_{TlR} subsume the according reaction rates as well as scaling factors for TsR and TlR concentrations.

Rates were optimized by fitting the model to the data from Experiments 1, 2, 3, and 4 numerically using a downhill simplex algorithm and subsequent least squares minimization (**Figure 5.2**). First, the maturation rate k_{mat} was deduced using data from Experiment 3. Second, parameters affecting the transcription reaction and mRNA decay (k_{ts}, m_{ts}, δ_{mRNA} and k_{TsR}) were optimized to fit measured mRNA levels (Experiment 4). Finally, using the resulting optimized transcription parameter set as initial values, all 8 free parameters (transcription and translation, without k_{mat}) were optimized simultaneously to fit all measured GFP levels (Experiments 1 and 2).

kinetic parameters $nM\backslash min^{-1}$	k_{ts}	m_{ts}	k_{tl}	m_{tl}	k_{mat}
	18.2	8.5	16.1	65.8	0.2
	k_{TsR}	δ_{TlR}	m_{TlR}	δ_{mRNA}	
	$1.1 \cdot 10^{-2}$	$4.5 \cdot 10^{-3}$	$6.3 \cdot 10^{-5}$	$4.5 \cdot 10^{-4}$	

Table 5.1 Parameter values of the ODE model as obtained by a fit to the experimental data.

Subsequent to parameter optimization, a Bayesian ensemble of parameters was created using a Markov Chain Monte Carlo approach (50000 steps in log space). Acceptance rate was at about 49.3 % resulting in 24669 accepted parameter vectors. Values of individual parameters of the Bayesian ensemble exhibit very small variances, indicating that a stable and robust optimum was reached by optimization (data not shown). See **Table 5.1** for the median parameter values and [104] for a comparison to values obtained from literature and a discussion. Model development, optimization, and parameter ensemble building has been performed using the SloppyCell [115, 116] and R software [117].

5.2.2 The PNFL Model

The PNFL model is realized as a semi-continuous model (Section 4.1.3), i.e. fuzzy logic systems are used to calculate changes of values for each species of the system. The PNFL model was created manually (**Figure 5.3**). This includes the definition of fuzzy sets for fuzzification and defuzzification and the definition of fuzzy logic systems. The model contains fuzzy logic systems for transcription, translation, TsR and TlR reduction, and mRNA decay. The maturation process (GFP to GFP*) was omitted, i.e. we do not distinguish between newly translated and maturated GFP. Transcription and translation fuzzy logic systems are modeled as simple activation reactions proportional to plasmid and TsR values, and proportional to mRNA and TlR. TsR is consumed by the transcription process, while TlR is decayed at a constant rate independent of the translation process. mRNA is decayed proportional to its concentration, i.e. subject to exponential decay. Initial plasmid concentrations are transformed prior to fuzzification using equation

$$y = \frac{\log_{10}(x/3.4) + 5}{5}$$

Thus, concentrations 3.4, 0.34, 0.034, 0.0034, 0.00034, and 0.0 nM are transformed to 1, 0.8, 0.6, 0.4, 0.2, and 0.0 (unit less). When fuzzified for the transcription fuzzy logic system, this results in the following linguistic descriptions and fuzzy values:

- very high template concentration ($< 0,0,0,0,0,1 >$)
- high template concentration ($< 0,0,0,0,1,0 >$)
- medium template concentration ($< 0,0,0,1,0,0 >$)
- low template concentration($< 0,0,1,0,0,0 >$)
- very low template concentration ($< 0,1,0,0,0,0 >$)
- absent template ($< 1,0,0,0,0,0 >$)

Thus, we represent each initial plasmid value (spanning several orders of magnitude) by an individual fuzzy set. The values of mRNA and GFP predicted by the model should be interpreted analogously, i.e. fuzzy sets representing those species correspond to concentration levels differing by orders of magnitude. Initial concentrations of mRNA and GFP are 0, initial concentrations of TsR and TlR are 1. To simulate Experiments 1, 2, and 4, the model is run for 100 iterations (**Figure 5.4**). Template DNA is inserted at iterations 0, 8, 16, 24, 32, and 40, representing the experimental addition times 0 min, 37 min, 73 min, 112 min, 153 min, and 187 min. TlR decay rates are chosen such that TlR concentrations reach 0 at iteration 50.

5.3 Selecting Models for Consumption and Decay

Some unspecified molecular components are necessary resources for transcription and translation processes. A reduction or exhaustion of these resources presumably causes the cessation of translation after about 4 h, and influences the slow saturation of mRNA levels. We stated three possible hypotheses which might explain the reduction of these resources: consumption only (hypothesis H.1), decay only (hypothesis H.2), consumption and decay (hypothesis H.3). Moreover, the hypothesis applying to the transcription resources might be different to the one applying to the translation resources. Our ODE and PNFL models can be used to further investigate these processes and help to decide for one of the hypotheses.

Both computational models presented in Section 5.2 have been already restricted to hypothesis H.1 for TsR and hypothesis H.2 for TlR, i.e. contain a consumption term for transcription components and a degradation term for translation components. Here, we describe the model selection process that led to these final ODE and PNFL models.

5.3.1 Model Selection using the ODE Model

Model selection for the ODE model was performed by including rate equations for consumption and decay reactions for both transcription as well as translation resources. Thus, by replacing the aforementioned Equations 5.4 and 5.5 by the following Equations 5.6 and 5.7 and by introducing additional parameters δ_{TsR}, m_{TsR} and k_{TlR}:

$$d/dt[TsR] = -\frac{k_{TsR} \cdot [TsR] \cdot [DNA]}{m_{ts} + [DNA]} - \frac{\delta_{TsR} \cdot [TsR]}{m_{TsR} + [TsR]} \tag{5.6}$$

$$d/dt[TlR] = -\frac{k_{TlR} \cdot [TlR] \cdot [mRNA]}{m_{tl} + [mRNA]} - \frac{\delta_{TlR} \cdot [TlR]}{m_{TlR} + [TlR]} \tag{5.7}$$

The additional parameters were optimized simultaneously with other model parameters as described in Section 5.2. Thereby, the influence of an actual translation process to the amount of the translation components was found to be marginal compared to the proposed translation-independent decay. The rate of the consumption process k_{TlR} always had its optimum very close to or at zero after multiple optimizations and thus hypotheses H.1 and H.3 compromising a consumption process can be rejected for translation resources. Analogously, hypotheses H.2 and H.3 compromising a transcription-independent decay of transcription resources can be rejected as optimizations always yielded a δ_{TsR} at zero.

5.3.2 Model Selection using the PNFL Model

PNFL models were created according to each hypothesis and mRNA and GFP kinetics were simulated using experimental settings similar to those of available experiments. The simulated qualitative kinetics were visually compared to measured kinetics. If the kinetics of a model contradicted the observed kinetics, the model and the according hypothesis were be rejected.

Transcription Process. We first investigated the influence of transcription resources consumption and/or degradation to the mRNA levels. Therefore we created three different models representing the transcription process. Each of these models was simulated using initial template DNA concentrations of 3.4 nM, 34 pM and 340 fM for 300 iterations (**Figure 5.5**, top):

- Transcription Model 1 (according to hypothesis H.1) includes the TsR consumption process as defined in **Figure 5.3**. This PNFL model predicts identical mRNA saturation levels and shifted saturation times proportional to template DNA concentration, i.e. mRNA saturates faster if high template DNA concentrations are used and vice versa.
- Transcription Model 2 (according to hypothesis H.2) includes a TsR decay process analogously defined to the TlR decay process as defined in **Figure 5.3**. This PNFL model predicts different mRNA saturation levels proportional to template DNA concentration at the same saturation time, i.e. higher template concentrations result in higher mRNA saturation levels.
- Transcription Model 3 (according to hypothesis H.3) includes both processes. Numeric values of centers of gravity of consequent fuzzy sets were halved to ensure that TsR reduction occurs at roughly the same speed as compared to transcription model 1 and 2. This PNFL model predicts different mRNA saturation levels and shifted saturation times.

The kinetics observed in Experiment 4 indicate shifted saturation times and thus an influence of consumption. Thus, hypothesis H.2 can be rejected and either hypothesis H.1 or H.3 may apply. As we have not observed mRNA saturation levels, we can not further distinguish between these two hypotheses.

Translation Process. We evaluated three models for the translation process which were defined analogously to the aforementioned transcription models. All three models include a TsR consumption process, i.e. transcription model 1. Using transcription model 3 results in similar kinetics. For all models, experimental settings of Experiment 2 were simulated, i.e. 3.4 nM template DNA added at different time points (**Figure 5.5**, bottom).

- Translation Model 1 (according to hypothesis H.1) includes a TlR consumption process analogously defined to the TsR consumption process as defined in **Figure 5.3**. This PNFL model predicts identical GFP saturation levels and shifted saturation times proportional to the addition times, i.e. later addition results in later saturation.
- Translation Model 2 (according to hypothesis H.2) includes the TlR decay process as defined in **Figure 5.3**. This PNFL model predicts different GFP saturation levels proportional to the addition times at the same saturation time, i.e. later addition results in lower saturation levels.
- Translation Model 3 (according to hypothesis H.3) includes both processes. Numeric values of centers of gravity of consequent fuzzy sets were halved to ensure that TlR reduction occurs at roughly the same speed as compared to translation model 1 and 2. This PNFL model predicts different GFP saturation levels and shifted saturation times, i.e. later addition results in lower saturation levels and later saturation times.

The kinetics observed in Experiment 2 indicate saturation levels proportional to addition time but no significant shift in saturation times. Thus, hypotheses H.1 and H.3 can be rejected and H.2 is the only remaining hypothesis.

5.4 Discussion

The *E. coli* like transcription/translation processes constitute a simple biological system. They occur in a reconstituted cell-free environment with a minimal number of molecular species that are necessary to allow transcription and translation and thus without influences of other molecular components that would be present in bacterial cells, e.g. proteases, RNAse, or regulatory factors. This simplifies the investigation of transcription and translation processes.

We developed an ODE model and optimized rate parameters, such that predicted GFP and mRNA kinetics fitted to the experimentally measured kinetics. Measurements were performed using template DNA concentrations spanning several orders of magnitude (from fM to nM), and resulting final GFP concentrations also spanned several orders of magnitude (from about 0.1 nM to 100 nM). The ODE model allows quite accurate predictions for all these concentration levels and predicts the observed reduction of GFP yield caused by a delayed template addition. The deviations we observed, especially for very low and very high GFP concentrations, might indicate that the model is inadequate in some details, but might also result from experimental noise, e.g. finite detector sensitivity, calibration errors, variations of actual DNA concentrations, and variations of actual addition times. Replicates for the experimental measurements were rarely available, so the expected influence of experimental noise could not be estimated [104]. Nevertheless, as the ODE model allows quite accurate predictions across several orders of concentrations and addition times, we can still assume that it is an adequate representation of the transcription/translation processes.

It is quite intriguing that the more general and thus more potent ODE model, namely the one comprising two different types of decay processes, was effectively reduced to a simpler model by a parameter optimization which evaluated parameter values solely by the fit of simulated to experimental data. Typically, one would expect that a more general model is able to reproduce (noisy) experimental data better, and that there is a trade-off between model simplicity and fit. However, in the case of the cell-free system, it seems that the chosen experimental settings and the acquired data were especially suited for model discrimination and support the reduced ODE model.

In contrast to the quantitative ODE model, which predicts the absolute value of GFP concentrations given initial template DNA concentrations, the PNFL models were explicitly designed to only provide a qualitative description of GFP time courses. Still, the PNFL models allow to predict essential properties of resource-dependent synthesis processes. For example, the transcription and translation processes where a product (mRNA or GFP) is synthesized from a template (DNA or mRNA), and where the processes are influenced by additional molecular components (transcription or translation resources). The PNFL models predict that:

- Consumption of components by a process can not account for different saturation levels of the product.

- Consumption of components leads to shifted product-saturation times, given different template concentrations.
- Process-independent decay of components causes different saturation levels of the product, given identical template concentrations and shifted addition times.
- Process-independent decay of components can not account for shifted product-saturation times.

Models including consumption and/or decay processes therefore exhibit specific qualitative behaviors. These can be compared to the qualitative behaviors of according experimental measurements to decide which models are adequate. The final PNFL model predicts that saturation levels but not saturation times depend on initial concentrations and that saturation levels depend on addition time. This corresponds to the experimental observations. Thus, the qualitative predictions of PNFL models are sufficient for hypotheses testing if the experimental observations exhibit qualitative different behavior.

Author's contribution The work presented here was performed during a joint project with Tobias Stögbauer and Joachim O. Rädler, and parts of it have been published in [104]. The author developed, implemented, and evaluated the computational models, while TS performed all experiments and participated in ODE model creation. The project was supervised by JR and Ralf Zimmer.

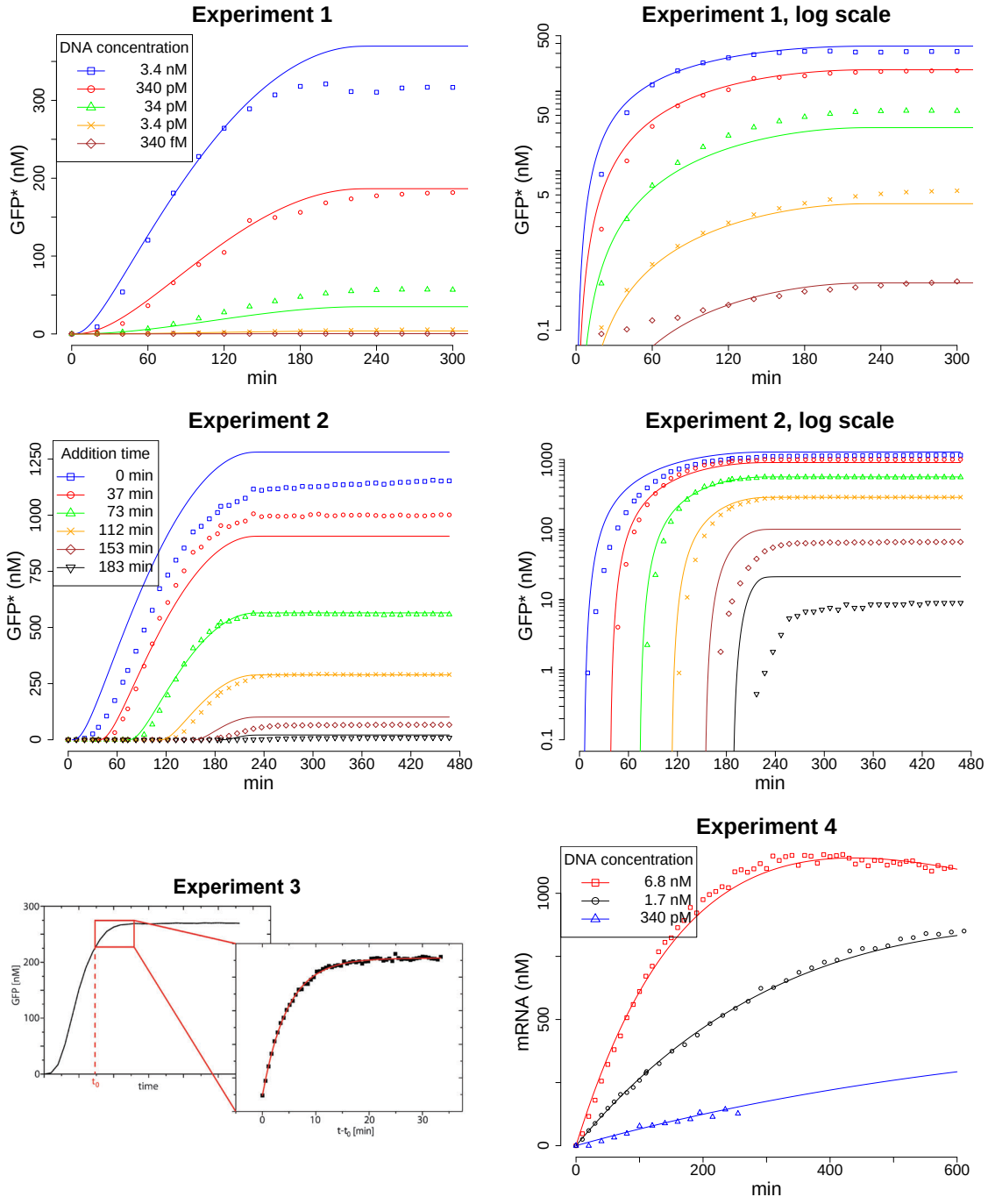
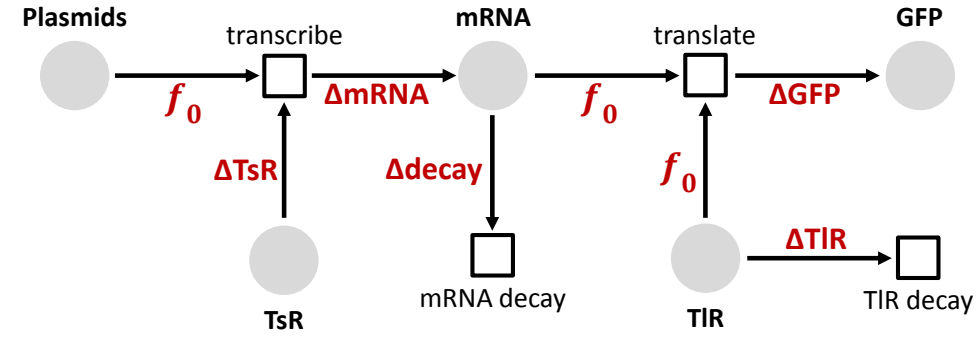


Figure 5.2 Experimentally measured kinetics of mature GFP (GFP*) and mRNA. All figures show experimental measurements (dots) and predictions of the ODE model (lines). Figures show GFP* kinetics for different initial template DNA concentrations (Experiment 1, top) or different template addition times after activation of the cell-free system (Experiment 2, center). The figure of Experiment 3 shows maturation kinetics for GFP after blocking of ribosomes at time t_0. This figure was taken from [104] (bottom left). Kinetics of GFP-mRNA for different initial template DNA concentrations were measured in Experiment 4 (bottom right).



TsR consumption **ΔTsR**

TsR	Plasmid	FS0	FS1	FS2	FS3	FS4	FS5
low		0	0	0	0	0	0
high		0	0.01	0.02	0.03	0.04	0.05

mRNA transcription **ΔmRNA**

TsR	Plasmid	FS0	FS1	FS2	FS3	FS4	FS5
low		0	0	0	0	0	0
high		0	0.01	0.02	0.03	0.04	0.05

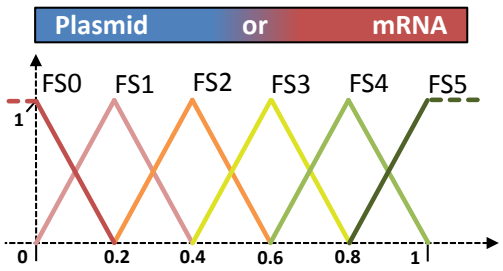
GFP translation **ΔGFP**

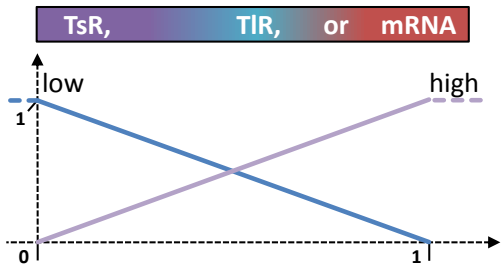
TlR	mRNA	FS0	FS1	FS2	FS3	FS4	FS5
low		0	0	0	0	0	0
high		0	0.01	0.02	0.03	0.04	0.05

mRNA decay **Δdecay**

mRNA	
low	high
0	0.01

Fuzzification





TlR decay **ΔTlR**

TlR	
low	high
0.2	0.2

Figure 5.3 PNFL model of the cell-free transcription/translation system. The Petri net (top) defines the connectivity of the species. Fuzzy logic systems or the zero function f_0 are assigned to arcs. The definitions of the fuzzy logic systems (bottom left) depict input species and according fuzzy sets (column and row headings) as well as the centers of gravity for consequent fuzzy sets. As height defuzzification is used, the shapes of consequent fuzzy sets have no influence on defuzzification and are not shown. Fuzzy sets used for fuzzifications (bottom right) are triangular and equally spaced in the interval [0,1]. First and last fuzzy set extend to (negative) infinity at height 1.

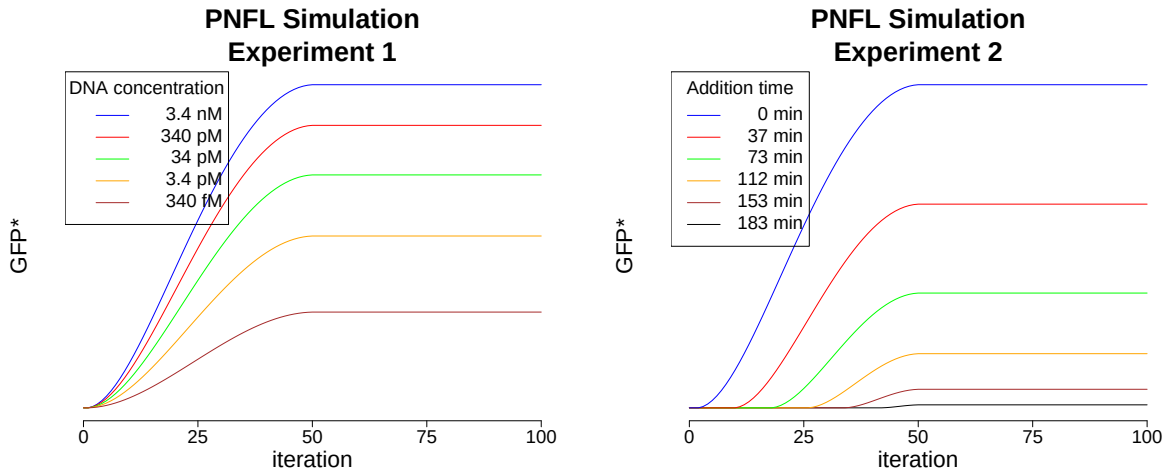


Figure 5.4 GFP kinetics simulated using the PNFL model. The PNFL model reproduces the qualitative behavior of the GFP kinetics as observed in Experiments 1 (left) and 2 (right). When simulating Experiment 1 using the five different initial plasmid concentrations, five clearly separated saturation levels are obtained after about 50 iterations. When simulating Experiment 2, where the same amount of initial plasmid is added at six different time points, clearly separated saturation levels and saturation after about 50 iterations are obtained.

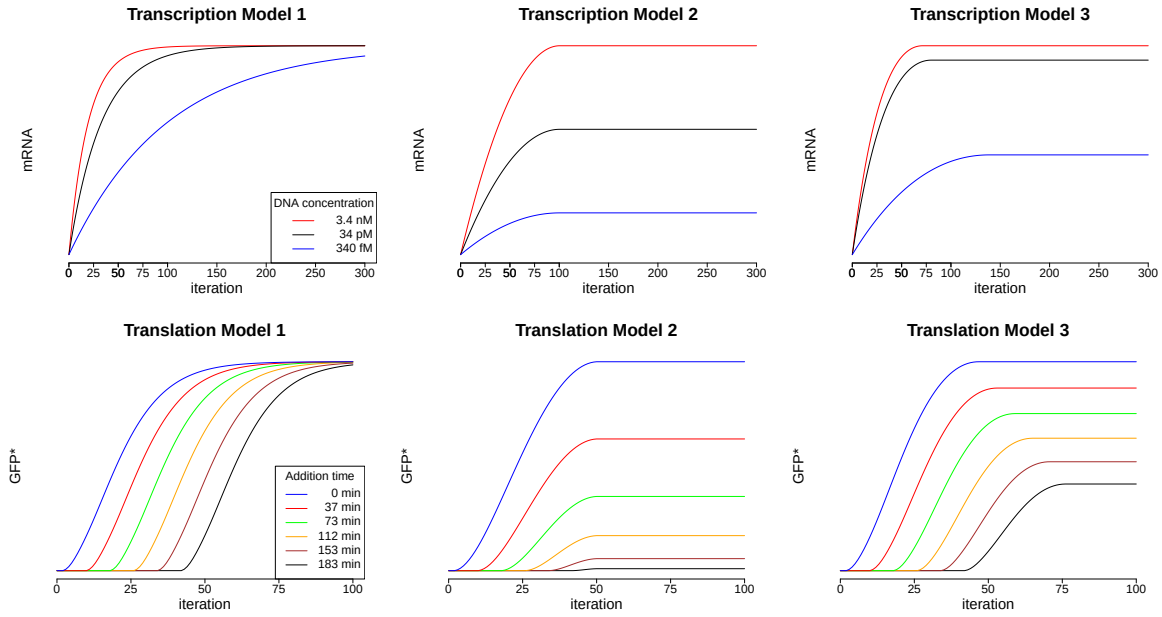


Figure 5.5 Comparison of mRNA and GFP kinetics for different transcription/translation models. The top row shows mRNA kinetics according to transcription models 1 (TsR consumption, left), 2 (TsR decay, center), and 3 (consumption and decay, right). Simulations with initial template DNA concentrations of 3.4 nM, 34 pM and 340 fM were performed. Experiment 4 (**Figure 5.2**) indicates shifted mRNA saturation times, which contradicts the kinetics simulated using model 2. Thus, this model and the according hypothesis can be rejected. The bottom row shows GFP kinetics according to translation models 1 (TlR consumption, left), 2 (TlR decay, center), and 3 (consumption and decay, right). We simulated initial template DNA concentration of 3.4 nM added after 0 min, 37 min, 73 min, 112 min, 153 min, and 187 min. Experiment 2 (**Figure 5.2**) indicates that GFP saturation times should be independent of addition time. Only model 2 shows this qualitative behavior, thus hypotheses H.1 and H.3 can be rejected.

Part II

Reverse-Engineering of PNFL Models

Chapter 6

A Genetic Algorithm for Reverse-Engineering

An aspect of reverse-engineering of biological systems or networks is the procedure of deriving effector-target relationships between entities, i.e. the underlying network structure, given experimental data and eventually additional prior knowledge. Reverse-engineering can be performed using dynamic models. Candidate dynamic models are built and used to simulate experiments. The simulated data is compared to experimental measurements. In general, one assumes that a good fit of simulated data to experimental data is an indicator that a created model is an adequate description of the biological system. Models are adjusted and simulations and evaluations are repeated until the fit of simulated and experimental data can not be further improved. Reverse-engineering can be done manually, but only for very small systems and a manageable amount of experimental data. Typically, reverse-engineering involves application of computational reverse-engineering algorithms.

To show that PNFL models can be successfully reverse-engineered, this section describes an approach based on a genetic algorithm [105]. The genetic algorithm was developed by Robert Küffner especially to infer PNFL models for the DREAM4 *in silico* network reconstruction challenge [118, 119]. The task of the DREAM4 challenge was to reverse-engineer artificial gene regulatory networks. The presence of direct, directed gene regulatory interactions between several virtual transcription factors should be inferred based on data from simulated experiments. These interactions represent regulatory effects of transcriptions factors (effectors) to the expression of their targets, which are other transcription factors.

The evaluation of the DREAM4 contest showed that our team was the best performing of 29 participants in the size 10 network reconstruction challenge ([105], Section 6.2.1). This already demonstrates that the described genetic algorithm together with the PNFL modeling technique is a successful and quite competitive reverse-engineering approach. In this thesis, the results of RK are supplemented by additional evaluations of the genetic algorithm to demonstrate its prediction quality on larger networks of up to 120 entities (Section 6.2).

The following sections describe the genetic algorithm (Section 6.1) and present evaluations of prediction performance (Section 6.2) that the author has performed based on DREAM4 reference networks as well as PNFL references. We do not present all details of the genetic algorithm, but

focus on those that clarify properties of the algorithm that are relevant for the methods presented in the following Sections 7 and 8. These properties are:

- The genetic algorithm is non-deterministic. Thus, repeated runs typically create similar but non-identical models.
- Each performed mutation affects only a small part of a model, i.e. one or two interactions, or a single entity.
- The applied simulated annealing technique allows for the acceptance of suboptimal mutations and suboptimal models.

These properties will be discussed further below (Section 6.3).

6.1 The Reverse-Engineering Algorithm

Genetic algorithms are search heuristics used for optimization of arbitrary search problems in high dimensional spaces [120, 121]. They mimic natural evolutionary processes. They are basically comprised of two operations that mimic mutations and crossovers. These operations are repeatedly applied to a population of potential solutions of the search problem. Here, these solutions correspond to concrete PNFL models that should explain the reference experimental data.

The presented genetic algorithm reverse-engineers models based on given reference experimental data sets. These data sets provide expression measurements for the entities of the system under various experimental conditions (Section 6.2.1). The entities of the system are known beforehand. The genetic algorithm only reverse-engineers the interactions of the system.

The algorithm initially creates a population of empty PNFL models, i.e. without interactions (**Figure 6.1**). During several hundred generations, the models of the population are repeatedly mutated (Section 6.1.2). After each mutation, the models are used to simulate the available experiments. Simulated and reference data sets are compared to evaluate the fitness of each model (Section 6.1.3). Whether a mutation is accepted or rejected is decided based on an acceptance probability (Section 6.1.4). Finally, a network of direct, directed interactions is derived from the best scoring PNFL model (**Figure 6.2**).

The structure and functionality of created PNFL models is strongly restricted (Section 6.1.1). The most relevant aspects of this restrictions are that all fuzzy logic systems have to be taken from a predefined collection and that only simple AND-, OR-, and MEAN-like totalizations based on the generalized mean are allowed. The fuzzy logic systems of the predefined collection represent individual effects of a single effector to a single target. They mimic simple activating and inhibiting effects of different strengths. The restriction of allowed fuzzy logic systems and totalizations massively reduces the search space (Section 6.3). Still, the restricted PNLF models are sufficiently powerful to capture the experimentally observed systems behavior and thus allow successful reverse-engineering of reference networks (Section 6.2).

```
1: procedure REVERSE-ENGINEER(D,I)
2:     P ← replicate(I, n_m)                          % Copy initial model n_m times
3:     repeat
4:         T ← T_0                                    % Reset temperature
5:         S ← Σ_{m∈P} σ(m)/n_m                       % Store average model score, Section 6.1.3
6:         for j ← 1, n_g do                          % Iterate for n_g generations
7:             for all m ∈ P do
8:                 m' ← mutate(m)                     % Mutate a model, Section 6.1.2
9:                 if accept(m', m, D, T) then        % Evaluate the mutated model, Section 6.1.3
10:                    P' ← P' ∪ m'
11:                else
12:                    P' ← P' ∪ m
13:                end if
14:            end for
15:            P ← P'
16:            T = T − T_0/n_g                        % Decrease the temperature
17:        end for
18:    until |S − Σ_{m∈P} σ(m)/n_m| < ε               % Check for convergence
19:    m_best ← argmax_{m∈P}(σ(m))                    % Get the best model m of P according to its score
20:    G ← convert_to_graph(m_best)                   % see Figure 6.2
21:    return G
22: end procedure
```

Figure 6.1 Pseudocode of the reverse-engineering algorithm. A reference network is reverse-engineered based on data set D using an initial model I.

```
1: procedure CONVERT_TO_GRAPH(m)
2:     for all p ∈ Places(m) do
3:         V ← V ∪ {v_p}                              % For each place, add a vertex to set V
4:     end for
5:     for all (p_e, p_t) ∈ Places(m) × Places(m) do
6:         if p_e is an effector of p_t in m then
7:             E ← E ∪ {(v_{p_e}, v_{p_t})}           % For each effector/target pair, add an edge to set E
8:         end if
9:     end for
10:    return G = (V, E)
11: end procedure
```

Figure 6.2 Pseudocode for converting a PNFL model to a directed graph. A PNFL model m is converted to a directed graph G.

6.1.1 Valid PNFL Models

All PNFL models created during execution of the genetic algorithm follow the definition given in Section 4 and are further restricted as described here.

Structural restrictions Each transition has exactly one output place and each place is output place of exactly one transition.

$$\forall t \in T : |(t)\bullet| = 1$$
$$\forall p \in P : |\bullet(p)| = 1$$

A place must not be input place and output place of the same transition.

$$p \in \bullet(t) \Rightarrow p \notin (t)\bullet$$
$$p \in (t)\bullet \Rightarrow p \notin \bullet(t)$$

Zero, one, or several places may be input places to a transition. A place can be input place to zero, one, or several transitions.

$$|\bullet(t)| \in \mathbb{Z}_{\geq 0}$$
$$|(p)\bullet| \in \mathbb{Z}_{\geq 0}$$

Thus, we have a one-to-one correspondence of places and transitions. We denote a place and its corresponding transition by using the same index: p_i is the single output place of t_i. All processes that affect a single place p_i are represented by its transition t_i.

Functional restrictions The zero-function f_0 is assigned to each input arc.

$$\forall (p,t) \in A : W((p,t)) = f_0$$

If a transition t_i has no input places, the zero-function f_0 is assigned to its output arc.

$$|\bullet(t_i)| = 0 \Rightarrow W((t_i, p_i)) = f_0$$

Otherwise, a function of the following form is assigned to the output arc to place p_i:

$$|\bullet(t_i)| \geq 1 \Rightarrow W((t_i, p_i)) = -\omega \cdot M(p_i) + \omega \cdot r_i \cdot GM_p(\bar{y}_1, \ldots, \bar{y}_n)$$

No other functions may be assigned to arcs. $M(p_i)$ is the current marking of place p_i. GM_p is the generalized mean with parameter p which controls the type of cooperative effect (Section 4.1.2). Parameter p must be chosen from a predefined collection of values (e.g. Section 6.2.2). Multiple effects to the same target are totalized by this generalized mean function. I.e. the generalized mean is applied to one or more $\bar{y}_j = fls_j(M(p_j))$, which are defuzzified output values of fuzzy logic systems. No nesting of multiple totalization functions is allowed.

The effect of an effector to a target is described by a single fuzzy logic system $\bar{y}_j$. Thus, there is a single fuzzy logic system assigned to each effector-target pair. All fuzzy logic systems have a single premise only. They use product inference and height defuzzification. A collection of fuzzy logic systems is predefined (e.g. **Figure 6.3**). All fuzzy logic systems must be chosen from this collection. Thus, candidate antecedent and consequent fuzzy concepts, as well as candidate rule bases are predefined.

Parameter $\omega \in (0,1]$ is a predefined update ratio. The same ω is used for all output arc functions. Parameter $r_i \in \mathbb{R}_{\geq 0}$ is a factor used to apply perturbation effects. It is individually adjusted according to experimental setups. Per default, r_i is 1.

6.1.2 Mutation Operations

During each iteration, each model of the population is mutated either by a simple mutation or by a recombination. Simple mutations affect single models and are performed without reference models. Recombination effects affect a single model using another model as reference. Which type of mutation is performed is randomly chosen according to a probability distribution P_{mut}. The probability $P_{mut}(\theta)$ of selecting a specific mutation of type θ is proportional to the acceptance rate of this type of mutation:

$$P_{mut}(\theta) \propto max(b_\theta, \alpha(\theta, t_{mut})/\pi(\theta, t_{mut}))$$

where $\pi(\theta, t_{mut})$ is the number of performed mutations of type θ during the last t_{mut} mutations that were performed during algorithm execution, and $\alpha(\theta, t_{mut})$ is the number of accepted mutations. Thus, simple mutations and recombinations that were successful in past iterations are preferentially selected. Parameter b_θ is used to ensure that $P_{mut}(\theta)$ is nonzero. Parameter t_{mut} specifies the number of past mutations that are tracked.

Simple mutations

M1: Randomly exchange a single effect. Randomly choose one of the genes with equal probability. Choose one of its current effectors randomly with equal probability. Replace the assigned fuzzy logic system by any other fuzzy logic system. Choose the other fuzzy logic system randomly with equal probability from the predefined collection of fuzzy logic systems.

M2: Optimize a single effect. Randomly choose one of the genes with equal probability. Choose one of its current effectors randomly with equal probability. Iteratively replace the currently assigned fuzzy logic system by another one and evaluate the fitness of the model (Section 6.1.3). Repeat this for all possible fuzzy logic systems. Keep the fuzzy logic system which results in the best fitness.

M3: Add a single effector. Choose a gene g_0 randomly with equal probability. Choose any other gene g_1 with probability proportional to a preference function $pf(g_1, g_0)$. Add g_1 as effector to g_0. Iteratively chose a fuzzy logic system and evaluate the fitness of the model.

Repeat this for all possible fuzzy logic systems. Keep the fuzzy logic system which results in the best fitness.

M4: Remove a single effector. Randomly choose one of the genes with equal probability. Iteratively remove one of the effectors of this gene and evaluate the fitness of the model. Add the removed effector again using the original fuzzy logic system. Repeat this for all effectors. Finally, remove the effector whose removing resulted in the best-fitting model.

M5: Randomly exchange totalization parameter. Choose a generalized mean function assigned to an output arc randomly with equal probability. Choose any other parameter p randomly with equal probability from the predefined set of totalization parameters.

M6: Replace effector or target. Choose a gene g_0 randomly with equal probability. Do one of the following with equal probability:

Replace one of g_0's effectors Choose one effector of g_0 randomly with equal probability. Denote this effector g_1. Choose any other gene g_2 randomly with with probability proportional to a preference function $pf(g_0, g_2)$. Remove g_1 as effector of g_0 and add g_2 as effector to g_0. Choose any fuzzy logic system with equal probability.

Replace one of g_0's targets Choose one target of g_0 randomly with equal probability. Denote this target g_1. Choose any other gene g_2 randomly with probability proportional to a preference function $pf(g_1, g_2)$. Remove g_0 as effector of g_1 and add g_0 as effector to g_2. Choose any fuzzy logic system with equal probability.

Recombination mutations

R1: Copy a single effect. Select any other model m_1 randomly with equal probability from the population of models. Choose a gene g_0 randomly with equal probability. Choose one of its effectors g_1 in m_0 randomly with equal probability. This g_1 must also be an effector of g_0 in m_1. Replace the according fuzzy logic system in m_0 by the fuzzy logic system present in m_1.

R2: Add a single effector. Select any other model m_1 randomly with equal probability. Choose a gene g_0 randomly with equal probability. Choose one of its effectors g_1 in m_1 randomly with equal probability. This g_1 must not be an effector of g_0 in m_0. Add g_1 as effector to g_0 in m_0. Use the same fuzzy logic system as in m_1.

R3: Remove an effector. Select any other model m_1 randomly with equal probability. Choose a gene g_0 randomly with equal probability. Choose one of its effectors g_1 in m_0 randomly with equal probability. This g_1 must not be an effector of g_0 in m_1. Remove g_1 as effector to g_0 in m_0.

R4: Copy a totalization parameter. Select any other model m_1 randomly with equal probability from the population of models. Choose a generalized mean function assigned to an output arc in m_0 randomly with equal probability. An according output arc must also be present in m_1. Replace the totalization parameter of m_0 by the totalization parameter of m_1.

R5: Replace effector or target. Select any other model m_1 randomly with equal probability. Choose a gene g_0 randomly with equal probability. Do one of the following with equal probability.

Replace one of g_0's effectors Find an effector of g_0 in m_0 which is not an effector of g_0 in m_1. Denote this effector g_1. Find an effector of g_0 in m_1 which is not an effector of g_0 in m_0. Denote this effector g_2. These g_1 and g_2 must not be the same. In m_0, remove g_1 as effector of g_0 and add g_2 as effector to g_0. Use the same fuzzy logic system as in m_1.

Replace one of g_0's targets Find a target of g_0 in m_0 which is not a target of g_0 in m_1. Denote this target g_1. Find a target of g_0 in m_1 which is not a target of g_0 in m_0. Denote this target g_2. These g_1 and g_2 must not be the same. In m_0, remove g_1 as target of g_0 and add g_2 as target to g_0. Use the same fuzzy logic system as in m_1.

If a mutation can not be performed successfully, e.g. if an effector should be removed but the chosen gene has no effector, or if the mutation would create an invalid model, another mutation is chosen instead.

We omit a detailed description of the preference function $pf(g_i, g_j)$. Basically, it assigns a high preference value to an unordered pair $\{g_i, g_j\}$ if previous mutations of g_i have significantly affected g_j or *vice versa*. Thus, a high value can only be assigned to $\{g_i, g_j\}$ if there is already a directed path of interactions connecting these two entities. The preference score is defined such that is always nonzero. We omit the description of mutations that assign, modify, or remove perturbation effects r_i that are relevant for time course experiments (Section 6.2.1). These mutations are performed similar to those presented above.

6.1.3 Simulation and Scoring

The simulation of each experiment is performed for a predefined number of iterations using the simultaneous firing rule. Initial states of entities are taken from the respective reference experiments. In case of knock-out or knock-down experiments, parameters r_i of the functions assigned to output arcs are modified according to the individual experimental setups. They are used to knock-out or knock-down genes either by setting the proposed new expression level to zero (knock-out), or by multiplying the proposed new expression level by a knock-down factor in the interval $(0, 1)$.

Simulated experimental data is compared to the reference data sets to evaluate the model's fitness. Measurements are taken from simulated time courses according to experimental setups such that they correspond to the reference measurements. First, the weighted correlation of si-

mulated data to reference data is calculated across all experiments for each entity individually.

$$\text{Weighted mean:} \quad avg(\mathbf{X}^{g_j}, \mathbf{W}) = \frac{\sum_{i=1}^{n} W_i \cdot X_i^{g_j}}{\sum_{i=1}^{n} W_i}$$

$$\text{Weighted variance:} \quad var(\mathbf{X}^{g_j}, \mathbf{W}) = \frac{\sum_{i=1}^{n} W_i \cdot (X_i^{g_j} - avg(\mathbf{X}^{g_j}, \mathbf{W}))^2}{\sum_{i=1}^{n} W_i}$$

$$\text{Weighted covariance:} \quad cov(\mathbf{X}^{g_j}, \mathbf{Y}^{g_j}, \mathbf{W}) = \frac{\sum_{i=1}^{n} W_i \cdot (X_i^{g_j} - avg(\mathbf{X}^{g_j}, \mathbf{W})) \cdot (Y_i^{g_j} - avg(\mathbf{Y}^{g_j}, \mathbf{W}))}{\sum_{i=1}^{n} W_i}$$

$$\text{Weighted correlation:} \quad \rho(\mathbf{X}^{g_j}, \mathbf{Y}^{g_j}, \mathbf{W}) = \frac{cov(\mathbf{X}^{g_j}, \mathbf{Y}^{g_j}, \mathbf{W})}{\sqrt{var(\mathbf{X}^{g_j}, \mathbf{W}) \cdot var(\mathbf{Y}^{g_j}, \mathbf{W})}}$$

where the sums are taken over all available measurement values of all data sets. $X_i^{g_j}$ and $Y_i^{g_j}$ are simulated values and reference values of a single entity. Each simulated and experimental measurement value is weighted by a factor $W_i \in \mathbb{R}_{\geq 0}$ to account for

- different number of measurements per experiment
- relevance of experiments
- relevance of individual measurements

For example, high W_i's can be used to increase the impact of knock-out experiments to the correlation, or small W_i's can be used to decrease the impact of repeated steady state measurements of time courses.

The model's fitness is quantified as a function of the average weighted correlation $\overline{\rho}$ across all entities and a penalty for model complexity:

$$\sigma(m) = \frac{log(1 - \overline{\rho} \cdot \overline{\rho})}{\sqrt{1 + s(m)}}$$

where the model size $s(m)$ corresponds to the weighted number of effector-target pairs. The weighting allows to influence the desired model size. The model's fitness is always negative. The higher the weighted correlation and thus the fit to experimental data, the smaller is $\sigma(m)$.

6.1.4 Simulated Annealing and Acceptance Probability

After each mutation, a model is evaluated as described above. After evaluation, it is decided whether the performed mutation is accepted or whether it should be undone. The probability of accepting the model m' is

$$P_{accept}(m') = \begin{cases} 1 & \text{if } \Delta_\sigma(m') < 0 \\ e^{-\frac{\Delta_\sigma(m')}{\bar{\Delta}} \cdot T^{-1}} & \text{else} \end{cases}$$

where $\Delta_\sigma(m') = \sigma(m') - \sigma(m)$ is the score difference between mutated and original model. The average score difference $\bar{\Delta}$ is calculated from the last t_Δ score differences calculated during algorithm execution.

The mutated model has a higher fitness if its score is smaller than the score of the original model. In this case, a mutation is always accepted, i.e. $P_{accept}(m') = 1$. Otherwise, the acceptance probability is influenced by the current temperature T. The higher the temperature, the more probable it is that a suboptimal model is accepted nevertheless. The temperature T is decreased by a constant value of $\frac{T_0}{n_g}$ after each generation, thus a linear cooling schedule is applied.

6.2 Evaluation of the Genetic Algorithm

We have evaluated the genetic algorithm on several artificial networks of different sizes and different sources. The evaluation was performed to estimate the prediction quality and runtime for several different reference network sizes. Two sources of reference networks have been used. First, artificial reference networks and simulated reference data sets provided for the DREAM4 *in silico* size 10 and size 100 challenge [122]. Second, self-produced networks and simulated data sets using PNFL models. In the following, we describe reference networks and data sets in more detail.

6.2.1 DREAM4 *in silico* networks

The task of the DREAM4 *in silico* challenge was to reverse-engineer artificial gene regulatory networks [122]. For this purpose, several data sets of simulated experimental measurements based on five reference networks of 10 genes and five reference networks of 100 genes were provided by the organizers.

Experimental setups and types of measurements All experiments were simulated for the same unknown total time interval and the unit of time is not specified. The number of simulated measurements and therefore the size of the data sets available for participants, depends on the type of experiment. At the beginning of each simulated experiment, the system is in a steady state. These steady states are specific for each reference and are the same for all experimental setups. Simulated measurements of expression levels for all genes were available. Protein concentrations were not included in the data set. After simulation, additive and multiplicative noise was added to all simulated measurements, including the initial states. The simulated expression levels were scaled to be in the interval $[0, 1]$. Four different types of experiments were performed: time course, knock-out, knock-down and multifactorial experiments:

Time course data sets Measurements from 20 time points are available. Plus a noisy steady state at time point 0. The time intervals between measurements are equal. A constant perturbation was applied to about one third of genes from time point 1 to time point 9. This perturbation is removed after time point 9. Which genes are perturbed is not known. A perturbation corresponds to a factor that is multiplied to the maximal transcription rates of affected genes. Perturbation factors are specific for each gene and are unknown. This type of experiment mimics perturbation experiments, e.g. heat exposure or chemical treatment.

Knock-out data sets Only one measurement of the final states is available. Each experiment corresponds to a single knock-out of a gene. Each gene is knocked-out once. The knocked-out gene is known. A knock-out is realized at time point 1, i.e. the system is initially in its steady state and reacts to the knock-out. The maximal transcription rate of the knocked-out gene is set to zero. This type of experiment mimics gene silencing, e.g. by RNA interference.

Knock-down data sets Knock-down experiments are similar to knock-out experiments, except that the maximal transcription rate of a knocked-down gene is multiplied with 0.5 instead of zero.

Multifactorial data sets Only one measurement of the final states is available. The maximal transcription rates of all genes were perturbed slightly across the whole time interval. The strengths of the perturbations are much lower than in time course experiments. The perturbation factors are not known.

Network types and available data We used the five size 10 and the five size 100 networks for our evaluations and derived additional networks based on the DREAM4 size 100 networks (**Table 6.1**).

DREAM4 size 10 networks are five reference networks each consisting of 10 genes. For each network, 5 time course, 10 knock-out, 10 knock-down, and 10 multifactorial data sets were provided by the DREAM4 organizers. Thus, 130 data-points per gene were available.

DREAM4 size 100 networks are five reference networks each consisting of 100 genes. For each network, 10 time course, 100 knock-out, and 100 knock-down data sets were provided by the DREAM4 organizers. Thus, 400 data-points per gene were available.

DREAM4 size 100 networks reduced by 1 layer were derived by the author based on the size 100 networks. Thereto, for each DREAM4 size 100 network the set of genes without outgoing interactions was identified. These genes and all ingoing interactions were removed from the network and from all data sets. This resulted in five reference networks of various sizes. We denote this type of networks as DREAM4 layer-1 networks. On average, 276 data-points per gene were available.

DREAM4 size 100 networks reduced by 3 layers were derived by iteratively performing the aforementioned procedure three times to get even smaller networks. We denote this type of networks as DREAM4 layer-3 networks. On average, 244 data-points per gene were available.

6.2.2 Reference Networks Based on PNFL Models

Random PNFL models were created as references and used to simulate reference data. These PNFL models follow the specification provided in Section 6.1.1. The model creation procedure

	DREAM				PNFL			
	size 10	layer-3	layer-1	size 100	size 30	size 60	size 90	size 120
genes	10	22	38	100	30	60	90	120
interactions	14	43	64	204	51	86	137	180

Table 6.1 Reference network statistics. The table provides the average number of genes and interactions of the five reference networks that were available or created for each setting.

is described in the following. A number of randomly chosen effectors was assigned to each gene based on the in-degree distribution derived from DREAM4 layer-1 networks:

$$(P(in-degree=0),\ldots,P(in-degree=5)) = (0.22,\ 0.36,\ 0.19,\ 0.12,\ 0.06,\ 0.04)$$

Each effector-target relation could be described by either a weak, medium, or strong activating, or a weak, medium, or strong inhibiting effect. These effects are described by appropriate rule bases (**Figure 6.3**). The effects were assigned randomly, thus the same effector typically has different effects to different targets. The same fuzzy concept was used for all antecedents and consequents. Its fuzzy sets were defined as:

FS_0 Trapezoid-like fuzzy set, unbounded to the left, $mp = 0$, $r = 0.2$.

FS_1 Triangle-like fuzzy set, $l = 0$, $mp = 0.2$, $r = 0.5$.

FS_2 Triangle-like fuzzy set, $l = 0.2$, $mp = 0.5$, $r = 1$.

FS_3 Trapezoid-like fuzzy set, unbounded to the right, $l = 0.5$, $mp = 1$.

The totalization parameters used for the generalized mean functions were randomly chosen from $\{-5, 1, 5\}$. The update ratio ω was 0.65 for all transitions.

weak activator			
FS_0	FS_1	FS_2	FS_3
0	0	0.2	1

medium activator			
FS_0	FS_1	FS_2	FS_3
0.2	0.2	0.5	1

strong activator			
FS_0	FS_1	FS_2	FS_3
0	0.5	1	1

weak inhibitor			
FS_0	FS_1	FS_2	FS_3
1	1	0.5	0.2

medium inhibitor			
FS_0	FS_1	FS_2	FS_3
1	0.5	0.2	0

strong inhibitor			
FS_0	FS_1	FS_2	FS_3
1	0.5	0	0

Figure 6.3 Rule bases for random PNFL models. The rule bases represent activating and inhibiting effects of different strengths. The according fuzzy logic systems are piecewise linear functions that calculate new expression values for the target entity. The tables specify the mapping of antecedent fuzzy sets to the centers of gravity of the consequent fuzzy sets.

Each PNFL model was simulated 10 times for 20 iterations in each case starting with random initial states to find the steady states of the models. If no gene was found to be oscillating in any of the 10 time courses, then one or more different steady states have been found for each gene. Then one of the time courses was selected randomly and its final states of all genes were used as

initial states for the subsequent creation of reference data sets for the reverse-engineering runs. If at least one gene was found to be oscillating in any time course, then for each gene the final states of all 10 time course were averaged and the average value was used as initial state. Thus, the initial states of non-oscillating genes correspond to their steady states and the initial states of oscillating genes are somewhere in between the maximal and minimal value of the oscillations.

Experimental setups and types of measurements Time course data sets as described in Section 6.2.1 were produced. Thereto, perturbation effects were assigned to 1, 2, or 3 genes per experiment. The number of target genes and the target genes themselves were randomly chosen. Each individual perturbation effect was randomly chosen from $\{0, 0.25, 0.5, 1.5, 1.75, 2\}$. Knock-out and knock-down experiments were produced as described in Section 6.2.1.

Network types and available data Random PNFL models of different sizes were produced (**Table 6.1**).

Size 30 random networks are five PNFL reference networks each consisting of 30 genes. For each network, 10 time course, 30 knock-out, and 30 knock-down data sets were simulated. Thus, 260 data-points per gene were available.

Size 60 random networks are five PNFL reference networks each consisting of 60 genes. For each network, 10 time course, 60 knock-out, and 60 knock-down data sets were simulated. Thus, 320 data-points per gene were available.

Size 90 random networks are five PNFL reference networks each consisting of 90 genes. For each network, 10 time course, 90 knock-out, and 90 knock-down data sets were simulated. Thus, 380 data-points per gene were available.

Size 120 random networks are five PNFL reference networks each consisting of 120 genes. For each network, 10 time course, 120 knock-out, and 120 knock-down data sets were simulated. Thus, 440 data-points per gene were available.

6.2.3 Reverse-Engineering Parameter and Evaluation Criteria

Reverse-engineering was performed using the genetic algorithm described in Section 6.1.1. The parameter for all reverse engineering runs were:

- Number of models in each population: $n_m = 10$.
- Number of generations performed before convergence is checked: $n_g = 500$.
- Minimal score difference necessary to continue algorithm: $\varepsilon = 1$.
- Initial temperature for simulated annealing: $T_0 = 0.2$.
- Weights assigned to experimental data: time courses 1, knock-outs 8, knock-downs 6, multifactorial 4.
- Other parameters: $t_{mut} = 4000$, $t_\Delta = 4000$, $b_\theta = 0.1$ if θ is a simple mutation, $b_\theta = 0.05$ if θ is a recombination.

The reference networks are known and predicted models are compared to those to check whether interactions have been successfully predicted. An interaction is successfully predicted if there is an according effector-target relationship present in a model. For each predicted model, the following values were computed:

Recall The percentage of reference interactions that are contained in the prediction.

Precision The percentage of reference interactions of all interactions contained in the prediction.

Overall CPU time The CPU time elapsed from initialization of the population to convergence.

Total number of generations The total number of generations performed until convergence.

As the genetic algorithm is non-deterministic (Section 6.3), repeated reverse-engineering runs typically predict different models. Thus, reverse-engineering was repeated 10 times to get an ensemble of predictions. The prediction performance is assessed across all predicted networks of all reference networks (**Figure 6.4**).

6.3 Discussion

We will not discuss the full details of the genetic algorithm that are presented in Section 6.1, i.e. the restriction of valid PNFL models, selection and implementation of mutation operations, scoring function design, simulated annealing procedure, etc. These have been conceived and implemented by RK [105]. The success of the genetic algorithm in the DREAM4 challenge justifies this design, which is, if not optimal, at least sufficiently good. Whether the design of the genetic algorithm or the chosen parametrization (Section 6.2.3) are optimal with respect to achievable prediction quality is of negligible importance for our discussion here as we do not make any statements about the achievable prediction quality. Instead, we focus our discussion on the influence of network size to prediction quality and on those properties of the genetic algorithm that are of importance for the methods presented in the following Sections 7 and 8.

Properties of the genetic algorithm

1. The genetic algorithm repeatedly mutates each model during generations. The type of mutation that should be performed as well as the entities that are influenced by the mutation are chosen randomly from various probability distributions (Section 6.1.2). Thus, the genetic algorithm is non-deterministic and performs a (guided) random walk in the search space, i.e. the set of all possible valid models.

2. The probability of accepting a mutation is calculated based on the score difference between modified and original model (Section 6.1.4). If the modified model has a better score than the original, a mutation is always accepted. But due to the applied simulated annealing procedure, mutations that result in a worse score might be accepted as well. Thus, the genetic algorithm can create suboptimal models during iterations and might terminate outside of an optimum.

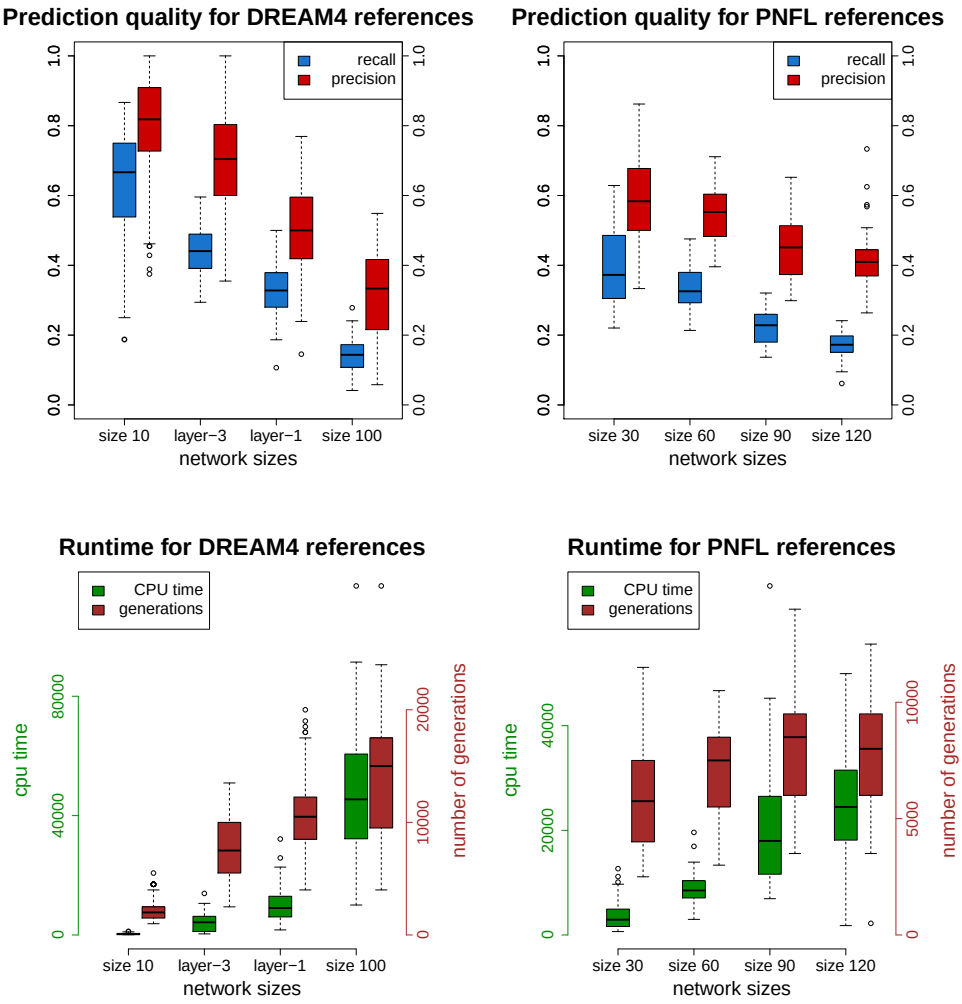


Figure 6.4 Genetic algorithm evaluation results. The top box-plots visualize the distributions of precision and recall for the 10 predictions of the 5 reference networks for each setting. Precision as well as recall drop significantly with increasing network size. The bottom box-plots visualize the according distributions of total CPU runtime in seconds and number of performed generations. CPU runtime increases due to the increasing number of generations but also due to increasing networks sizes, as the runtime needed for individual simulations increases.

3. All mutation operations affect only a small part of a model (Section 6.1.2), i.e. simple mutations M1-M4 affect a single effector-target relationship, M5 affects a single entity, and M6 affects two effector-target relationships. Recombination operations affect at most one model and affect only a small part of this model, i.e. R1-R3 affect a single effector-target relationship, R4 affects a single entity, and R5 affects two effector-target relationships. Thus, the genetic algorithm performs only small changes to models during generations.

Because of the first two properties, repeated reverse-engineering runs typically predict different models. The algorithm might terminate outside of an optimum, terminate in different local optima, or terminate in different global optima. Thus, predicted models can be similar, but may be quite different as well. The third property causes the genetic algorithm to perform only small steps through the search space, but no big leaps. Thus, it depends on the presence of paths of small, mostly beneficial mutations that lead to optimal models (see below for more details).

Prediction quality The evaluation of prediction performance shows that the genetic algorithm performs very well when predicting small networks (e.g. DREAM4 size 10 networks), but that prediction quality drops when network sizes increase (**Figure 6.4**). For DREAM4 size 10 networks, the reverse-engineered models contain on average about 64 % of all reference interactions, and about 80 % of predicted interactions were reference interactions. These values drop to about 15 % and 32 % for DREAM4 size 100 networks.

We see an essential reason for the decrease of prediction quality for larger networks in the massive increase of the search space that has to be searched by the genetic algorithm. The search space of the reverse-engineering algorithm corresponds to the number of possible valid models. For the model specifications and experimental settings used in out evaluations, the size of the search space $|H|$ can be calculated as follows:

$$|H| \approx \underbrace{7^{g(g-1)}}_{\text{a}} \cdot \underbrace{3^{g}}_{\text{b}} \cdot \underbrace{7^{gp}}_{\text{c}}$$

Term a is the number of different possible effect assignments for $g(g-1)$ effector-target pairs given seven possible effects, i.e. one of six fuzzy logic systems is assigned or there is no effect. Term b is the number of different possible parametrizations of generalized mean functions. Term c is the number of different possible assignments of p perturbators to g genes given six possible perturbation effects or no effect. For a network of $g = 10$ genes with $p = 5$ perturbation time courses, this search space is already immense and contains about $1.2 \cdot 10^{123}$ different models.

Obviously, it is highly unlikely that a reverse-engineering algorithm that would only create unrelated random guesses about model structure is able to create a good prediction. Still, we observe that our reverse-engineering algorithm performs surprisingly well when predicting small networks from such huge search spaces. (e.g. DREAM4 size 10 networks, **Figure 6.4**).

We presume that such a surprisingly good performance can only then be achieved, if models can be stepwise improved by a series of small mutations, i.e. by adding, removing, or modifying single effects, each of which are in general beneficial for the model's fitness.

As we have seen, our reverse-engineering algorithm performs only such small steps within the search space. After each mutation, the mutated model is evaluated and the mutation is only

accepted, if the model's fitness is sufficiently improved such that it outperforms the penalty for model size. Thus, most of the accepted small mutations necessarily improve fitness and can be seen as steps directed towards an optimum in the search space. The only exceptions might be some few unfavorable mutations which may be accepted due to the applied simulated annealing procedure.

Necessarily, there have to be paths of mostly beneficial mutations that lead from initial models to the final models that are similar to the reference networks, i.e. final models that contain a high fraction of reference interactions and a low fraction of non-reference interactions. This seems only then reasonable, if the addition of a single reference interaction or the removal of a single non-reference interaction is a beneficial mutation in general.

As naturally the reference interactions are not known beforehand, the genetic algorithm can not specifically test them. Instead, whenever the genetic algorithm performs a mutation that aims to add an effect, the respective effector-target pair is chosen randomly amongst all possible interactions. Thus, the probability that a reference interaction is chosen for addition to the model depends on the proportion of reference interactions among all possible interactions. The actual number of reference interactions is typically much smaller than the number of possible interactions. For example, in the DREAM4 size 10 reference networks about 14 interactions are contained on average. This corresponds to about 16 % of the 90 possible interactions. In contrast, in the DREAM4 size 100 reference networks about 204 interactions of 9900 possible interactions are contained. This corresponds to a proportion of only about 2 %.

So, we see that the probability that a reference interaction is chosen strongly decreases with network size, as the number of possible interactions increases as a quadratic function of the number of entities, while the number of reference interactions is linearly proportional to the number of entities (as typical for scale-free biological networks). Due to their strongly decreasing proportion, it becomes more unlikely that reference interactions are tested and added during the generations of the reverse-engineering runs. Thus, it is scarcely assessed whether adding a certain reference interaction would be beneficial for the current model.

We see this as the essential reason why the prediction quality significantly drops with network size, both in terms of precision as well as recall (**Figure 6.4**). The genetic algorithm often appears to be not able to find beneficial mutations during an iteration and terminates. One could strongly increase the number of performed mutations to ensure that reference interactions are tested frequently. But this would drastically increase the runtime of the genetic algorithm. Instead, we propose an iterative prediction procedure for the reverse-engineering of larger networks, that increases the probability that reference interactions are selected during the genetic algorithm's execution (Section 7).

Author's contribution The genetic algorithm was developed by Robert Küffner [105]. The author designed and performed the evaluations presented in Section 6.2.

Chapter 7

Iterative Prediction of Large Network Models

The evaluation of the reverse-engineering algorithm described in Section 6 showed that its prediction performance significantly drops with increasing number of genes. We hypothesized that this is caused by the massive increase of the search space size, i.e. the massive increase of possible models. The applied reverse-engineering algorithm is obviously not able to find good models within the huge model space in an acceptable time. It typically terminates after several hundred generations without finding sufficiently beneficial mutations. The increase of search space goes along with a significant decrease of the proportion of reference interactions amongst the set of possible interactions (reference interactions are those interactions that are actually present in the reference network). Due to the low proportion of reference interactions, they are rarely evaluated by the genetic algorithm, as the interactions affected by mutations are selected randomly.

A possible approach used to improve prediction quality of larger networks is a systematic incorporation of prior knowledge about network structure. Several groups have evaluated the influence of prior knowledge to prediction performance [86, 123, 124]. These approaches typically restrict the set of possible directed pairwise interactions a reverse-engineering algorithm might choose from to build a model of the system. The number of possible interactions increases as a quadratic function of the number of genes in a network. Thus, to significantly restrict the number of possible interactions, a comprehensive prior knowledge about interactions is required. Such comprehensive prior knowledge is often not available, cumbersome to obtain, and error prone.

We have developed an approach which uses scores that are derived from the already available experimental data as prior knowledge to restrict the huge search space. The main idea of the approach is to apply an iterative procedure which first trains preliminary PNFL models in a strongly reduced search space, then relaxes the search space and uses the preliminary models as starting points for further reverse-engineering. Hereby, the restriction of the search space is performed such that the proportion of reference interactions is significantly increased. This allows that the genetic algorithm evaluates these presumably beneficial interactions more frequently.

```
1: procedure PREDICT-ITERATIVELY(D,C)
2:     create an empty initial model m
3:     for all c ∈ C do
4:         K_c ← restrict_interactions(D,c)        % Restrict the set of allowed interactions
5:         m ← reverse-engineer(D,m,K_c)           % Reverse-engineer in the reduced model space
6:     end for
7:     m ← reverse-engineer(D,m)                   % Finally, reverse-engineer in the full model space
8:     return m
9: end procedure
```

Figure 7.1 Pseudocode of the iterative procedure. The iterative prediction procedure encapsulates the genetic algorithm presented in Section 6.1. A sorted collection of cutoffs $C = (c_1, \ldots, c_n)$ with $c_i > c_{i+1}$ is given as additional parameter next to the reference data D. The set of $g(g-1)$ possible interactions is restricted based on a scoring scheme (Section 7.2) and the given cutoffs. During each iteration, the genetic algorithm is only allowed to select from the restricted interaction set. After each iteration, the restricted interaction set is relaxed by selecting the next smaller cutoff.

7.1 Iterative Prediction Procedure

At first, the reverse-engineering is performed on a very restricted set of candidate interactions only. This way, a preliminary model of the system is trained. Preliminary models will already show similarities to the reference model although they might contain fewer interactions, especially only a subset of the reference interactions. After each convergence of the reverse-engineering algorithm, the restrictions are relaxed such that more interactions could be included into models. The reverse-engineering is repeated on the relaxed set using the previous model as initial model. Relaxation and reverse-engineering is iterated several times. During the final round of reverse-engineering the full search space is available and all possible models could be reached in principle by starting from the last preliminary model (**Figure 7.1**).

Restriction of interactions has to be done such that the proportion of reference interactions is increased in the set of candidate interactions. This can be done based on some prior guess about which interactions are actually reference interactions, for example based on prior information about interactions from yeast-two-hybrid experiments or other sources [86, 123, 124], or the restriction could be derived from the available experimental data sets as described in Section 7.2. There are three basic assumptions that have to hold if the iterative procedure should be successful:

1. The reverse-engineering algorithm can find an optimal model more easily within the restricted search space.
2. The resulting preliminary model is a better initial model for further optimization runs than a random or empty model.
3. A preliminary model can still be modified in subsequent iterations when the restrictions of the search space are relaxed.

The first assumption is rather trivial. The smaller the search space, the easier it can be searched for optimal models. This is supported by our evaluations in Section 6 where we have found

that our genetic algorithm performs significantly better when predicting smaller networks in accordingly smaller search spaces. The third assumption holds as the effector-target relationships, fuzzy logic systems, and other aspects of initial PNFL models can be freely mutated by the genetic algorithm. Furthermore, due to the applied simulated annealing approach and therefore the possibility that suboptimal mutations are accepted, any other possible model can be reached in principle starting from any initial model. This has already been shown in Section 6. To support the second assumption, we argue that:

Preliminary models that are already similar to the reference are preferred initial models, as they facilitate the prediction of further reference interactions. In general, one can state that some reference interactions are beneficial for models although their full reference-context is not established yet, i.e. the model yet misses other reference interactions due to the restricted search space or other reasons. Such reference interactions can be denoted as context insensitive interactions. Other reference interactions might only have beneficial effects to a model's fitness, if a context of other reference interactions is already established (context sensitive interactions). If reference interactions can be enriched using a preliminary model, parts of the necessary context might thereby be established. During subsequent iterations, this established context should facilitate the prediction of context sensitive reference interactions. Thus, models that are already similar to the reference are preferred initial models.

The preliminary models that result from predictions on the restricted search space are already significantly more similar to the reference than random models. This holds if the restricted search space contains reference interactions, and if the reverse-engineering algorithm predicts some or most of these interactions, while predicting relatively few non-reference interactions. The former can be achieved by a procedure that restricts the search space to a subset of interactions such that reference interactions are enriched within this subset. We present an appropriate procedure in Section 7.2. The latter can be seen from the evaluation results (Section 7.3) and is addressed in the discussion (Section 7.4).

7.2 Data-Driven Restriction of Candidate Interactions

The set of $g(g-1)$ possible directed interactions between g genes should be restricted such that reference interactions are enriched within the remaining candidate interactions. Therefore, we initially assess for each gene which other gene it might influence based on the available experimental data sets that later will be used for the reverse-engineering. I.e. we assess whether a direct or indirect interaction is present between a pair (g_A, g_B) of genes. We investigated two different scores. The first score $c(g_A, g_B)$ is the weighted absolute correlation of all measurements obtained from all available data sets:

$$c(g_A, g_B) = |\rho(X^{g_A}, X^{g_B}, W)| = \left|\frac{cov(X^{g_A}, X^{g_B}, W)}{\sqrt{var(X^{g_A}, W) \cdot var(X^{g_B}, W)}}\right|$$

where X^{g_i} are all measurements of gene g_i. Weighted covariance and weighted variance are described in Section 6.1.3. This score is symmetric, thus an interaction from gene g_A to gene g_B has the same score as the interaction from g_B to g_A.

The second score $z(g_A, g_B)$ is the maximal absolute z-score of g_B in knock-out and knock-down experiments of g_A:

$$z(g_A, g_B) = \max_{X^{g_B}}(\left|\frac{X^{g_B} - m^{g_B}}{\sigma^{g_B}}\right|) \tag{7.1}$$

where the maximum is evaluated over measured values of g_B in all knock-out and knock-down experiments where g_A is knocked-out or knocked-down. Value m^{g_B} is the mean steady state value of g_B, and value σ^{g_B} is the standard deviation of g_B's steady state value. The score $z(g_A, g_B)$ is asymmetric.

The weighted absolute correlation reflects whether there is a positive or negative correlation of expression levels or concentrations between two genes. A high correlation indicates that there might be an effector-target relationship between the two genes, or that they are at least co-regulated. The maximal absolute z-score indicates whether the knock-out or knock-down of a gene has a significant effect to the expression of another gene. This indicates that there is either a direct interaction or a series of interactions connecting the knocked-out and the influenced gene.

In DREAM4 size 100 networks, weighted absolute correlation as well as maximal absolute z-score of reference interactions are increased as compared to the respective scores of non-reference interactions (**Figure 7.2**). Both scores enrich reference interactions in high scoring subsets of interactions, but the effect is considerably more pronounced when using the maximal absolute z-score. The same holds for DREAM4 size 10 as well as layer-1 and layer-3 networks (data not shown).

7.3 Evaluation Results

We evaluated the iterative prediction procedure using the same settings as described in Section 6.2. The restrictions of interactions were based on the maximal absolute z-score. The sets of candidate interactions are derived by selecting all pairs of genes with a score greater than given score cutoffs. The applied cutoffs were 5, 3, and 2 (**Table 7.1**). Thus, three iterations and a final prediction without restrictions were performed. The restricted interaction sets contain on average only about 3 %, 6 %, and 14 % of all possible interactions. Reference interactions are significantly enriched in these sets and on average account for 53 %, 38 %, and 20 % of the contained interactions as compared to 2 % in the full set (DREAM 4 size 100).

The restricted interaction sets that are created during the iterative procedure can be seen as score based prediction of the network's structure (**Figure 7.2**, right). Hereby, all candidate interactions above a certain score could be classified as reference interactions and all other candidate interactions could be classified as non-interactions. This provides some baseline values for precision and recall. The predictions of the iterative procedure considerably outperform the predictions based on the applied score in terms of precision (**Figure 7.3**). The recall of the iterative

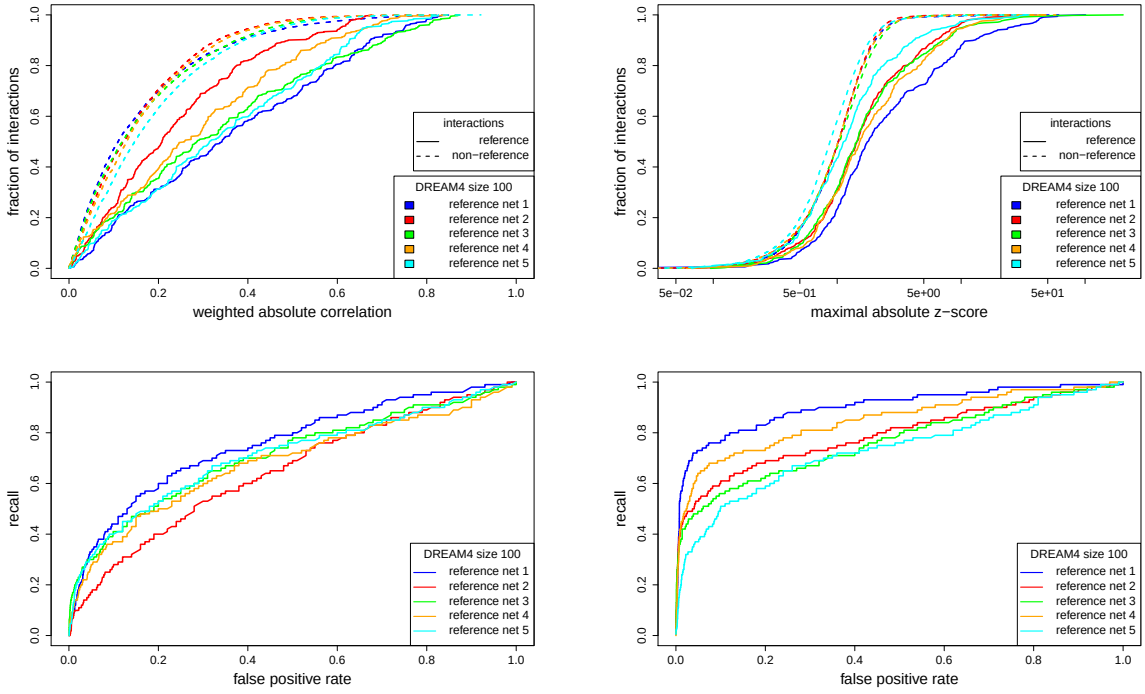


Figure 7.2 Evaluation of interaction scores. The cumulative distributions (top) and receiver operating characteristics (bottom) of weighted absolute correlations (left) and maximal absolute z-scores (right) for DREAM4 size 100 networks are shown. Reference-interactions (solid lines) have increased scores as compared to non-reference interactions (dashed lines). The average areas under the ROC curves are 0.698 (left) and 0.811 (right). Interactions were considered as undirected for the evaluation of weighted absolute correlation.

predictions is necessarily lower than the recall of score-based predictions. After the first iteration, the iterative predictions nearly achieved the theoretically possible recall, but recall increased only slightly in later iterations, i.e. when the search space is relaxed. Most reference interactions are found during the first iteration, where the most stringent cutoff is applied. In later iterations, still several interactions are added and interactions of the preliminary models are removed or modified (**Figure 7.4**). Thus, the preliminary models are not static but are repeatedly modified during iterations.

The total runtime of the iterative procedure including the final prediction on the full interaction set is similar to the runtime of the simple genetic algorithm (**Figure 7.5**). This holds for both total CPU time as well as the number of generations before termination. Although the number of candidate interactions strongly increases with each iteration (**Table 7.1**), the number of generations performed during the iterations only slightly increases (**Figure 7.6**).

The main result of the evaluations is that the predictions of the iterative procedure outperform the simple genetic algorithm in terms of precision as well as recall (**Figure 7.7**). For the smallest networks of up to about 20 genes (DREAM4 size 10, DREAM4 layer-3), precision and recall of the iterative procedure and the simple genetic algorithm are still comparable. For larger networks the iterative procedure significantly outperforms the simple genetic algorithm. Most importantly, precision as well as recall are nearly doubled for the largest networks (DREAM4 size 100, PNFL size 120).

7.4 Discussion

To evaluate a model, its score is derived by comparing the simulated data sets to the reference data sets. The better the fit of simulated and reference data, the higher the model scores. Each model is typically penalized for its size, for example by taking into account the number of interactions that are contained in the model. But scoring does not take into account the size, nor other properties of the search space. Thus, the score of a model is not influenced by restricting or relaxing the search space, as long as the model remains unchanged. From this follows that a model which is locally or globally optimal in the full search space must also be locally or globally optimal in any restricted search space, as long as it is contained in this search space. Following the basic assumption of reverse-engineering, i.e. that a good score of a model is associated with a higher proportion of reference interactions, we can conclude that reference interactions should be enriched in an optimal model that was created by applying the reverse-engineering algorithm on the restricted search space.

Although most reference interactions are excluded during the first iteration (**Table 7.1**), we observed that many of the allowed reference interactions are contained in preliminary models, and that reference interactions are enriched compared to non-reference interactions (**Figure 7.3**). Thus, many reference interactions are beneficial for preliminary models although the set of candidate interactions is restricted. It seems that these reference interactions can be included into models without prior establishment of a certain context, i.e. these are context insensitive interactions. If a reverse-engineering algorithm adds such a context insensitive interaction to the current model, it increases the score of this model in most cases. By repeatedly adding of context in-

sensitive interactions, the algorithm performs small beneficial steps through the model space and creates the preliminary models. In contrast, it is quite hard to predict context sensitive reference interactions, as their necessary context has to be established beforehand. Thus, such interactions will only be added if the model has already achieved the necessary context by other beneficial modifications. So we assume that the reverse-engineering algorithm is only then able to predict models that are similar to the reference, if most reference interactions are context insensitive and most non-reference interactions are either context sensitive or generally non-beneficial.

Although one can assume that the preliminary models are good initial models for the subsequent iterations of reverse-engineering, we observe only a slight increase of recall in later iterations. This might be due to three reasons. First, the proportion of reference interactions in the restricted interaction sets decreases in later iterations (**Table 7.1**). Thereby, the probability that these are tested by the genetic algorithm decreases as well. Thus, the original problem of the genetic algorithm stated at the beginning of this section might still persist. Second, the algorithm might have reached a local optimum during the first iterations and is not able to leave it in later iterations to find a better optimum, i.e. a model that contains more reference interactions. Third, it could be that adding an interaction from an effector to a target is only beneficial if the target reacts strongly to a knock-out or knock-down of the effector. Or *vice versa*, only if the target shows a strong reaction to an effector, the interaction between this effector-target pair can be predicted. Thus, the strict restriction of interactions might correspond to masking those interactions that can hardly be predicted by the applied reverse-engineering approach anyway. What is actually the case is subject of further evaluations.

Concluding, we can state that the iterative procedure allows for high-precision predictions of larger networks with sizes of up to a few hundred genes. Nevertheless, the prediction quality, especially the recall, still decreases with increasing network size, while the runtime increases strongly. Thus, the iterative procedure is not suited for predictions of networks with $\gg 100$ genes. The iterative approach can be easily extended to incorporate other prior knowledge about network structure. For example, if some candidate interactions should be excluded from all models due to prior knowledge, the restricted interaction sets can be adjusted accordingly during all iterations and the final prediction. Thus, if prior knowledge is available, it can be utilized to further increase prediction quality.

Author's contribution The author developed the iterative procedure. Further evaluations are currently performed, and a manuscript describing the iterative approach is in preparation.

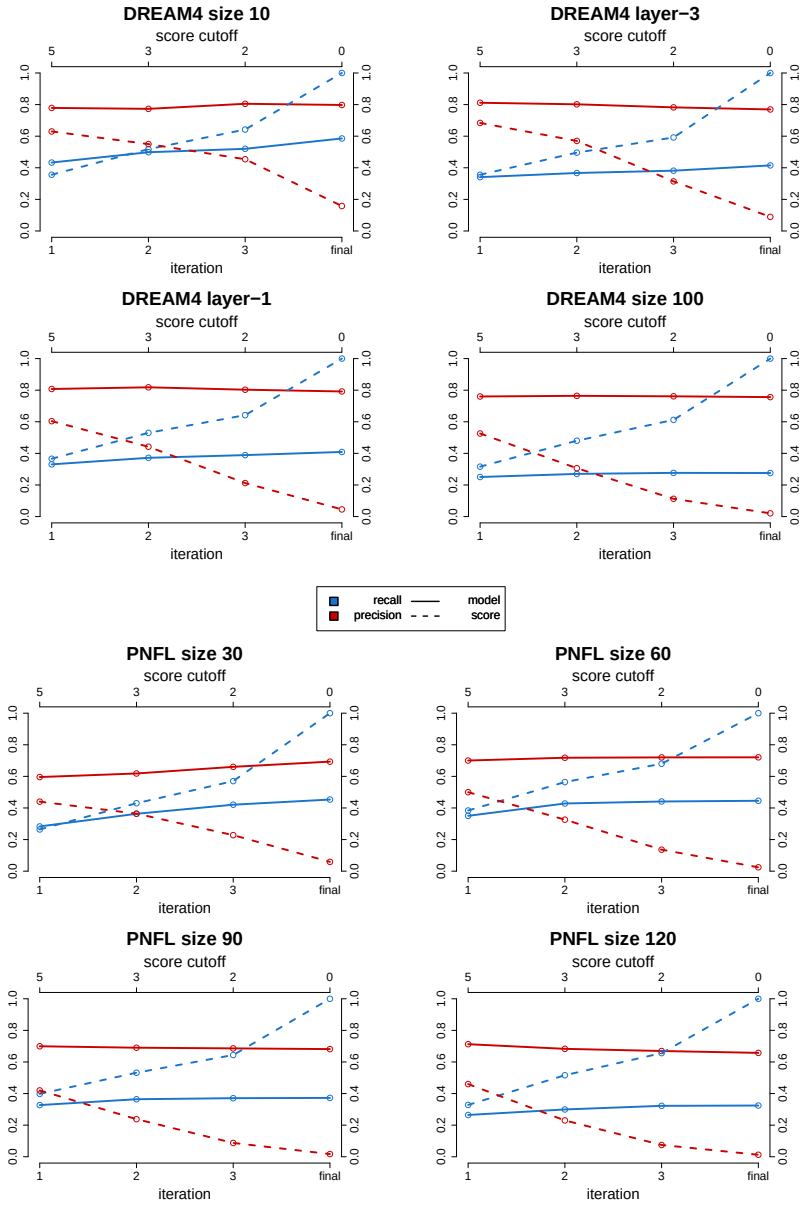


Figure 7.3 Comparison to score based predictions. These plots show the mean recall and precision values of the iterative procedure as well as those obtained by simply applying the restriction cutoffs to interactions (score-based prediction, Section 7.3). The mean recall of the score-based predictions can be seemingly lower than the mean recall of the iterative procedure. This is an artifact of taking the mean of predictions for different references (compare to **Figure 7.7**). We observe that the precision of score-based predictions is significantly lower compared to the model-based predictions for all applied cutoffs. Thus, the iterative procedure outperforms the score-based predictions with respect to overall prediction quality.

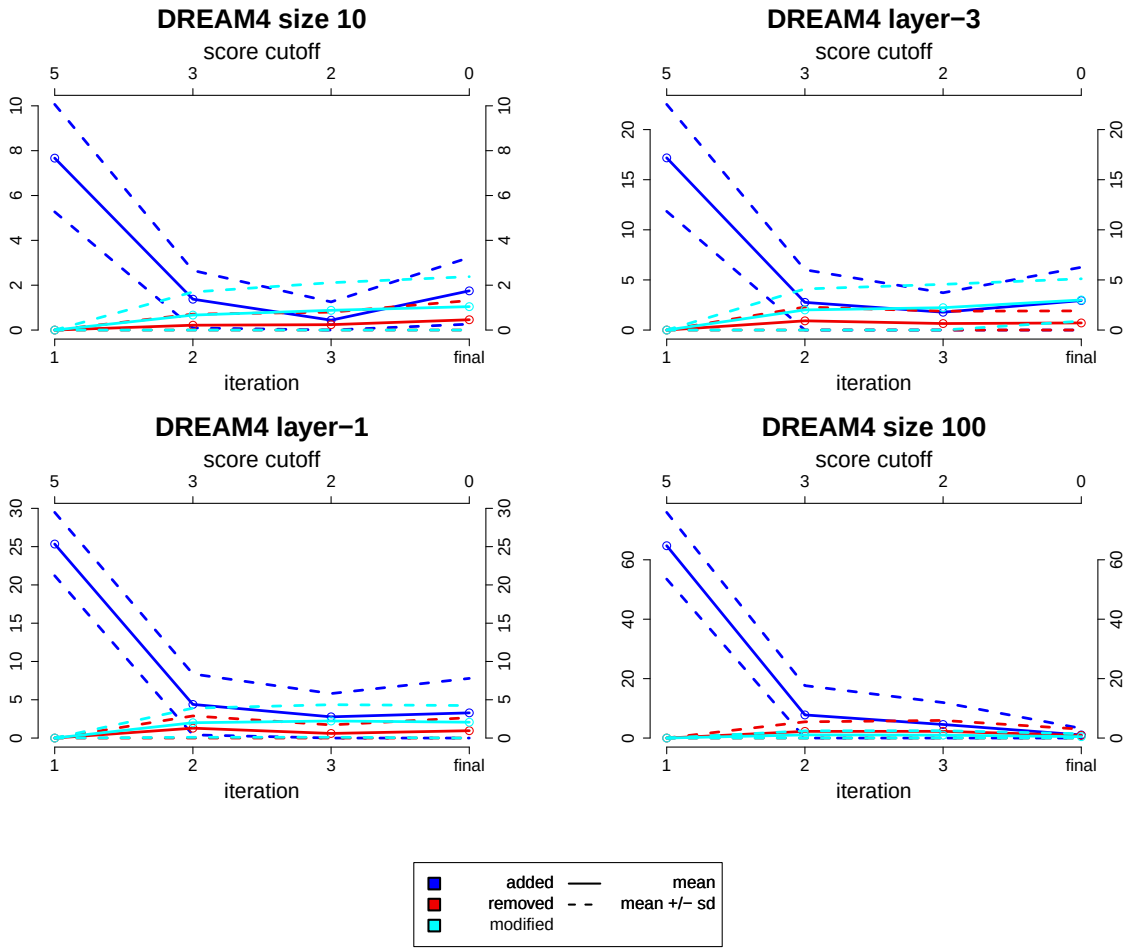


Figure 7.4 Modifications of models. The figures show the number of added, removed, and modified interactions at the end of an iteration compared to the initial model of each iteration. As at iteration 1 an empty initial model is used, no interactions can be removed or modified. Although most interactions are added during the first iteration, some are still added during later iterations. The preliminary models used as initial models are not static but modified by removing or modifying interactions. Modifying interactions corresponds to a change of the assigned fuzzy logic system or of the parameter of the generalized mean function that is assigned to the respective transition.

experimental setting	cutoffs			
	5	3	2	0
DREAM4 size 10	7.0 (4.4)	12.6 (6.9)	21.8 (9.8)	90 (14.2)
DREAM4 layer-3	21.4 (14.6)	40.8 (23.2)	84.6 (26.5)	480 (42.9)
DREAM4 layer-1	38.0 (22.9)	77.8 (34.3)	196.4 (41.6)	1406 (63.8)
DREAM4 size 100	120.8 (63.5)	325.4 (99.5)	1112.2 (125.0)	9900 (203.9)
PNFL size 30	29.2 (12.8)	59.2 (21.5)	127.4 (29.0)	870 (51.1)
PNFL size 60	66.8 (33.4)	152.0 (49.5)	440.4 (59.8)	3540 (86.3)
PNFL size 90	135.0 (56.7)	315.4 (75.0)	1029.8 (90.0)	8010 (137.7)
PNFL size 120	132.2 (60.8)	406.6 (93.5)	1589.2 (118.5)	14280 (179.9)

Table 7.1 Sizes of restricted interaction sets averaged over all five references that were available for each experimental setting. The average number of contained reference interactions is shown in parentheses. All interactions with a maximal absolute z-score above the given cutoffs are included in the restricted sets. For cutoffs 5, 3, and 2, there are on average 3 %, 6 %, and 14 % of possible interactions contained in the respective restricted sets. A cutoff of zero corresponds to the full set of possible directed pairwise interactions.

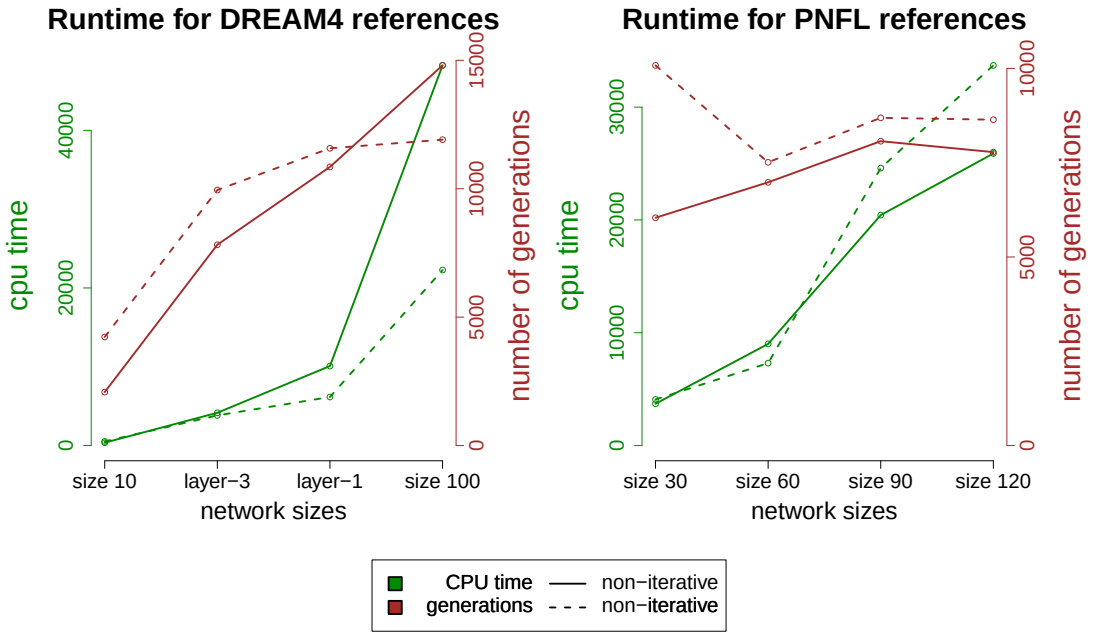


Figure 7.5 Comparison of runtimes. The plot shows the average CPU runtime in seconds and number of generations across all prediction runs for the according types of references. For the iterative procedure CPU time and number of generations were totalized across all iterations as well as the final prediction. Both CPU runtime and number of generations do not differ strongly between the iterative procedure and the (non-iterative) simple genetic algorithm.

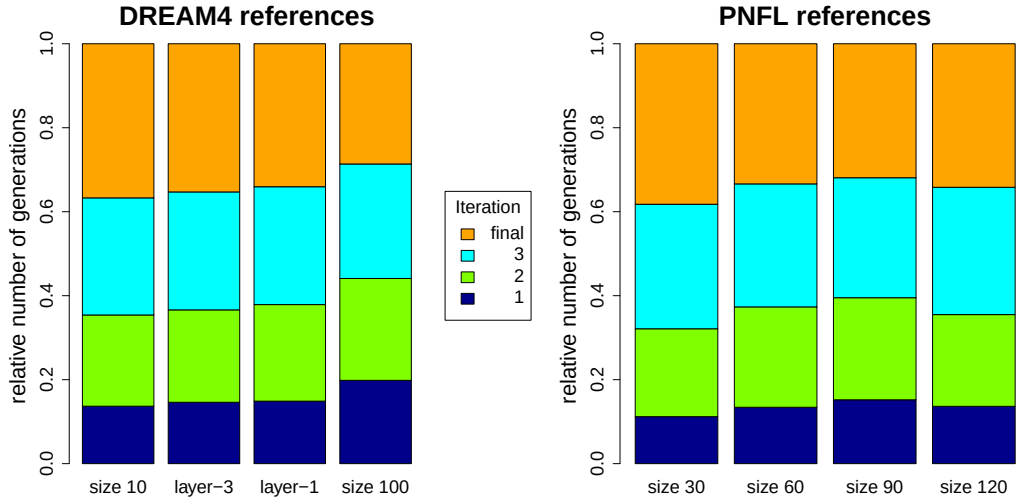


Figure 7.6 Relative number of generations during iterations. Although the number of candidate interactions strongly increases with each iteration (compare **Table 7.1**), the number of generations performed by the genetic algorithm only slightly increases. The relative number of generations is hardly influenced by the reference network size.

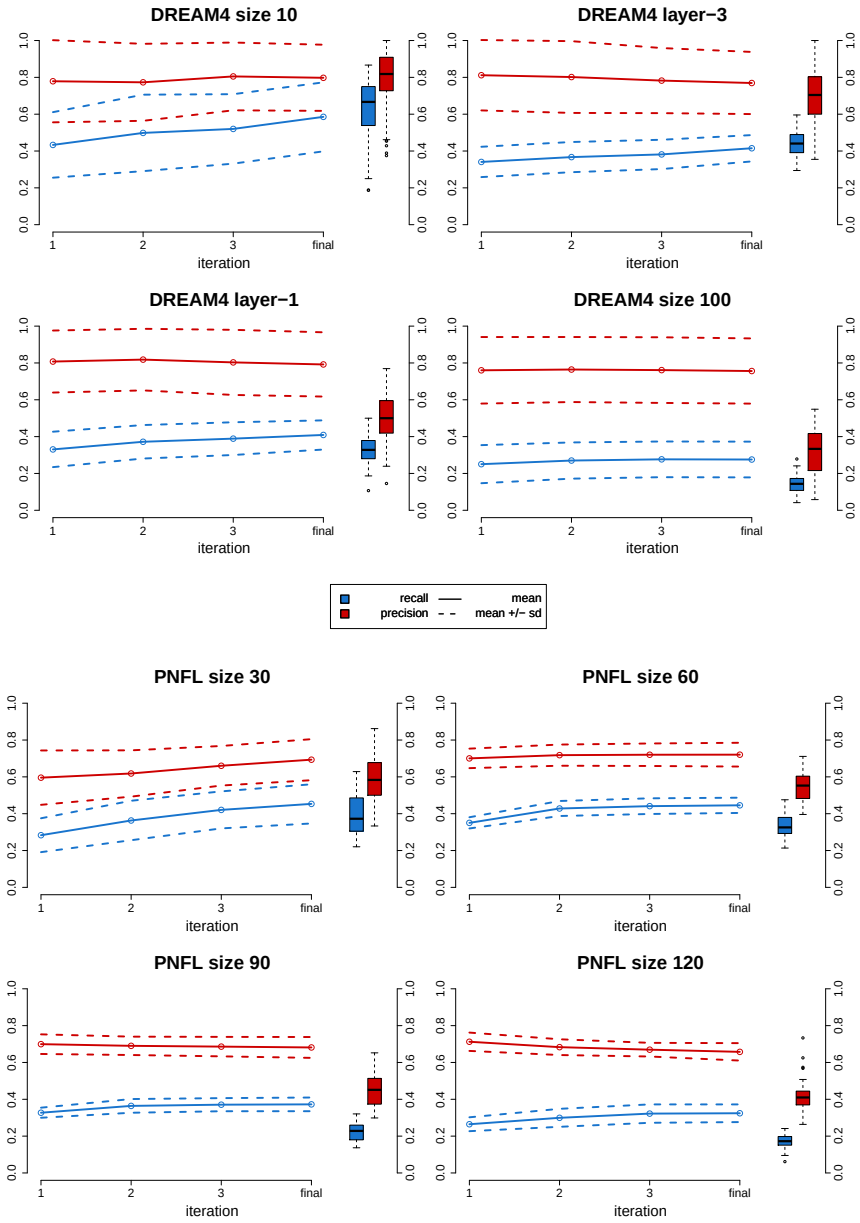


Figure 7.7 Evaluation of iterative predictions. Each of these plots summarizes the prediction qualities for the ten predictions that were performed for each of the five reference networks. Thus, means and standard deviations are calculated based on 50 models for each plot. Recall and precision of preliminary models after first, second, and third iteration, as well as of the final model are shown as lines. For comparison, the prediction quality of the simple genetic algorithm is shown as box-plot (compare to **Figure 6.4**). Especially for larger networks, the iterative procedure performs significantly better than the simple genetic algorithm.

Chapter 8

Ensemble Approach for Reverse-Engineering

The reverse-engineering algorithm introduced in Section 6 is based on random mutations of model parameters. Thus, this algorithm performs a non-deterministic optimization of PNFL models. Non-deterministic optimization is typically repeated several hundred or thousand times to collect high scoring networks [105, 125, 126]. In the case of reverse-engineering of dynamic models, high scoring networks are those networks that are derived from models that are able to reproduce the experimental data well. Most of these models are structurally different to each other, and none of them might be identical to the reference network. This is due to three fundamental reasons which might apply individually or jointly (adapted from [127], **Figure 8.1A**):

Representational The applied mathematical framework is not suited to represent the true regulatory relations, for example due to simplifications. In such a case, also the best scoring predicted network might not be identical to the true gene regulatory network.

Statistical Several different models reproduce experimental data equally well and thus lead to different acceptable hypotheses. Even if a derived network equals the true regulatory network, the associated model can not be distinguished from others by its fit to the data.

Computational The applied optimization algorithm might get stuck at high scoring local optima. Thus, resulting networks are sampled from suboptimal regions of the search space.

Nevertheless, it can be expected that interactions which are present in the reference network are enriched in high scoring networks. This follows from the basic assumption of reverse-engineering, i.e. that dynamic models that are able to reproduce experimental data well are adequate models, and that networks derived from these models are similar to the underlying biological networks [59].

If a confidence for individual effector-target relations should be derived by reverse-engineering, it is more promising to consider the frequencies of interactions in all high scoring networks than to select a single prediction. Several of such ensemble approaches have been proposed and have been found to be superior in terms of precision, recall and robustness ([128] and

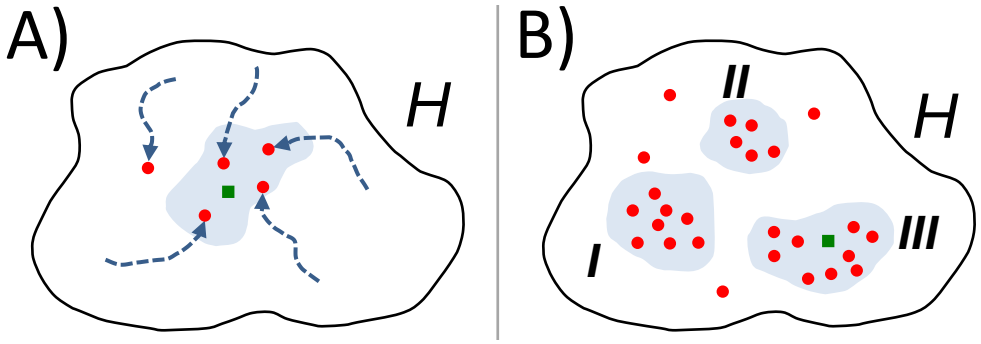


Figure 8.1 Motivation for ensemble averaging and its drawback. A) The search space *H* contains all networks that can be represented by the mathematical framework that is used for reverse-engineering. Depending on the available experimental data, the applied scoring functions, and the mathematical framework, several different model parametrizations, and therefore network structures, might score similarly. There might be no single optimum in *H*, as these different network structures are equally valid. Additionally, optimization procedures starting from different initial parametrizations might get stuck at local optima and create suboptimal predictions. If the applied framework is adequate, the reference structure is included in *H* and could be predicted by the optimization procedure. Otherwise, predicted high scoring networks should be at least similar to the reference. Here, all high scoring predicted networks are very similar to each other and to the reference. In such a case, the frequencies of interactions in all networks are reliable indicators for the confidence of an effector-target gene relation, thus applying ensemble voting is advisable.
B) Depending on the reference structure, the applied mathematical framework, and the available experimental data, several groups of topologically different high scoring networks might be predicted by a probabilistic reverse-engineering algorithm (blue areas *I*, *II* and *III*). Combining all of these structurally different networks by ensemble voting would obscure characteristics of individual groups of networks.
The search space *H* is symbolized by the black shape. Each point within the black shape symbolizes a unique network structure. Similarly high scoring networks are those contained in the blue area. The random walks of optimization procedures are symbolized by dashed arrows. The bases of these arrows correspond to initial model parametrizations and the final models are symbolized by red dots. The distance between red dots symbolizes the distance between network structures. The network structure which is identical to the reference network is symbolized by a green square. Figures are adapted from [127].

references therein). For example, voting schemes like majority voting, signed or unsigned voting, or weighted voting, can be applied to derive scores for each possible effector-target relation (**Figure 8.2**).

$$mv(A) = \begin{cases} 1 & \text{if } p > max(a,n) \\ -1 & \text{if } n > max(a,p) \\ 0 & \text{else} \end{cases}$$

$$p = |\{M \subseteq N | \forall_{m \in M}\, s_A^{(m)} = 1\}|$$

$$n = |\{M \subseteq N | \forall_{m \in M}\, s_A^{(m)} = -1\}|$$

$$a = |\{M \subseteq N | \forall_{m \in M}\, s_A^{(m)} = 0\}|$$

$$sv(A) = \frac{\sum_{m \in N} s_A^{(m)}}{|N|}$$

$$uv(A) = \frac{\sum_{m \in N} |s_A^{(m)}|}{|N|}$$

$$wv(A) = \frac{\sum_{m \in N} w^{(m)} \cdot s_A^{(m)}}{\sum_{m \in N} w^{(m)}}$$

Figure 8.2 Ensemble voting schemes. The set of predicted networks is denoted N. The sign of an interaction A in the network $m \in N$ is denoted $s_A^{(m)} \in \{-1, 0, 1\}$, where $s_A^{(m)} = 1$ describes an activating interaction, $s_A^{(m)} = -1$ describes an inhibiting interaction, and $s_A^{(m)} = 0$ if the interaction is absent in the network. $w^{(m)}$ is a weight assigned to network m. (Left) When using majority voting, a score of either 1, -1, or 0 is assigned to each interaction A according to whether this interaction is activating, inhibiting, or absent in the majority of networks. (Top center) The signed voting score $sv(A)$ can be seen as the average sign of an interaction A. For example, if the number of networks that contain A as activating interaction is larger than the number of networks that contain A as inhibiting, $sv(A)$ will be positive. It is 1 if all networks contain activating interaction A. A small absolute value of $sv(A)$ indicates that either most networks miss interaction A or that the number of activating and inhibiting occurrences is similar. (Bottom center) The unsigned voting score $uv(A)$ corresponds to the fraction of networks that contain interaction A, ignoring whether it is activating or inhibiting. (Right) Weighted voting is similar to signed voting, but here each network is weighted according to its score. Networks which are derived from models that reproduce the experimentally data better gain a higher weight.

These scores (weights) typically correspond to the frequency of an interaction in proposed networks. In the case of weighted voting, scores might range from 1 (high confidence interaction) via intermediate scores (low confidence interaction) to 0 (high confidence non-interaction), where "high confidence" is synonymous to "observed in most/few considered networks". High confidence interactions are apparently necessary to reproduce experimental data, as they are present in all high scoring networks. On the contrary, high confidence non-interactions might contradict experimental data, or their presence does not increase the models fit but only its complexity and thus is disfavored. Low confidence interactions constitute variable sub-regions of networks, i.e. their functionality seems to be beneficial, but might as well be realized by alternative interactions. Therefore, these interactions are only present in a subset of high scoring networks while alternatives are present in others.

8.1 Flaws of Ensemble Voting And How to Overcome Them

Ensemble voting can be an adequate technique if considered networks are sufficiently similar to each other. However, if networks differ strongly in overall or local topology (**Figure 8.1B**),

then ensemble voting "lead[s] to a meaningless blur of alternative structures" [128]. In the following, we will discuss this flaw of ensemble voting in more detail and motivate our approach to overcome it. The discussion is illustrated by the small gene regulatory network in **Figure 8.3**.

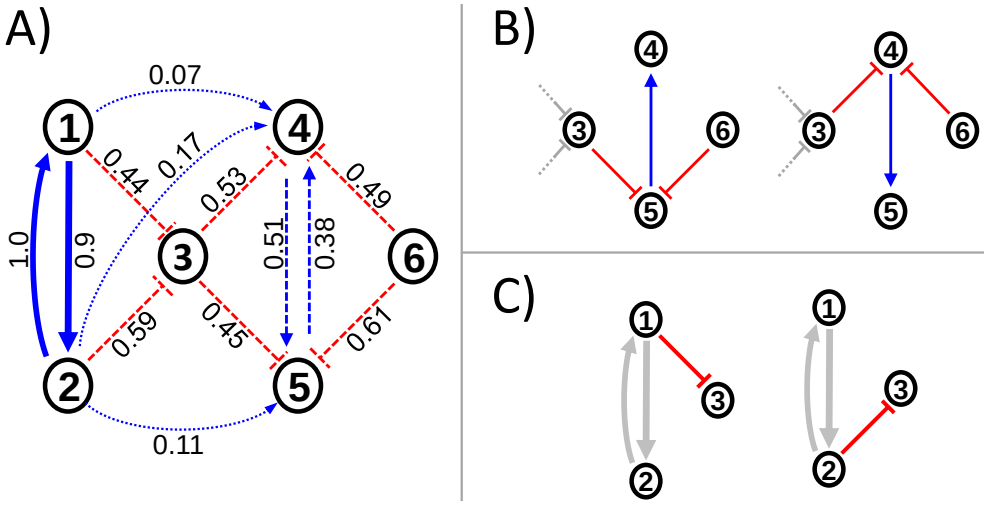


Figure 8.3 Illustration of an ensemble. An ensemble of several hundred predicted networks is created by calculating the frequencies of interactions. Reverse-engineering algorithms may produce suboptimal predictions, thus a certain amount of (random) variations in network topologies has to be expected. A) Ensemble average with annotated relative frequencies for activating (blue) and inhibiting (red) interactions. High confidence interactions (bold) are present in nearly all networks. High confidence non-interactions (dotted) are missing in most. Interactions present in subsets of networks have intermediate weights and are considered as low confidence interactions. B) Interactions connecting genes 3 to 6 constitute two characteristic sets (low confidence interactions). Either the left set of interactions or the right one is realized in predicted networks, but rarely a mixture of sets or subsets. The co-occurrence of these interactions is not apparent in the ensemble in A). C) Interactions affecting gene 3 are mutually exclusive, but do not co-occur with other interactions. They can occur in combination with any of the characteristic sets and constitute unspecific, highly variable sub-regions of networks.

Assume that several hundred or thousand networks with different topologies have been created by a reverse-engineering algorithm due to statistical, representational, and computational reasons as described before. The interactions derived from all these networks can now be divided according to their frequencies: first, interactions that are present in nearly all networks; second, interactions that are missing in nearly all networks; third, interactions that are present in subsets of networks only. When applying ensemble voting, the latter would constitute low confidence interactions.

Low confidence interactions can be further subdivided according to their mutual dependencies. First, interactions that always co-occur with one or more other low confidence interactions; second, variable interactions without co-occurrence relations. We define two interactions as co-occurring if the presence of one interaction is a reliable indicator for the presence of the other interaction and *vice versa*. In general, the presence of interactions is conditioned by the available experimental data, the chosen mathematical framework, and the applied reverse-engineering algorithm. Thus, the simultaneous presence of a pair of interactions seems to be necessary for a

required functionality, i.e. only then a network can be high scoring. Obviously, some networks do not contain the set of co-occurring interactions, as otherwise these interactions would not be of low confidence. In these networks, some competing set of low confidence interactions has to exhibit the otherwise missing functionality, and these interactions could be co-occurring as well. An example for competing sets of co-occurring interactions is given in **Figure 8.3B**. Interactions that do not co-occur with others constitute highly variable sub-regions of predicted networks. They typically arise whenever multiple effector candidates for a single target exist that can not be distinguished by the available data, and thus can be freely exchanged. Interactions arising from multiple effector candidates might have a redundant functionality, so they increase the complexity of a model without increasing its fit to experimental data. Thus, such interactions are often mutually exclusive, although not necessarily.

If ensemble voting would be applied using all predicted networks, mutual dependencies would be obscured as associated interactions become indistinguishable from unspecific, highly variable interactions, and meaningful information would be lost. Thus, we propose that networks should be grouped according to the contained sets of co-occurring interactions (characteristic sets) and that ensemble voting should be performed separately on each group. Thereby, the interesting common characteristics can be preserved, as co-occurring interactions would be enriched in the resulting group-ensembles.

In the following, we present an approach for identifying mutually dependent interactions from a set of network predictions, combining co-occurring interactions to characteristic sets, and grouping networks according to the presence of these characteristic sets. We show that group-ensembles derived by an ensemble voting are superior to the ensemble of all networks in terms of interpretability, and that co-occurring interactions are especially suited for experimental verification.

8.2 A Characteristic Interaction Set Extraction Approach

The approach consists of three subsequent steps: Calculation of interaction frequencies, derivation of scores for mutual dependencies, and finally grouping of networks. Due to the inherently high variability of networks caused by suboptimal predictions, a certain amount of *noise* has to be expected, i.e. redundant or missing interactions in any network. The input data is a set of high scoring networks predicted by a non-deterministic reverse-engineering algorithm, e.g. the reverse-engineering algorithm introduced in Section 6. Each of these networks has the same number of nodes, representing genes, and a variable number of interactions, each representing a regulatory influence of an effector-gene to a target-gene. During the following explanation signs of interactions (activating or inhibiting) are omitted for simplicity. The extension of the approach is straightforward and was applied for evaluations.

Step 1: Interaction frequencies Each interaction A is classified according to its relative frequency $f(A)$ in all networks as

1. high confidence interaction if its relative frequency is above a cutoff,

2. high confidence non-interaction if its relative frequency is below a cutoff,
3. and low confidence interaction otherwise.

Only low confidence interactions are of interest for further processing, as stated in the introduction. For each pair of low confidence interactions (A,B) the relative frequency of its co-occurrence $f(A,B)$ in all networks is calculated.

Step 2: Mutual Dependencies For each pair of low confidence interactions a score for two mutual dependency relations is calculated as follows:

$$s_{AND}(A,B) = \frac{f(A,B)}{\max(f(A),f(B))}$$
$$s_{EX}(A,B) = \min\left(\frac{f(A,\neg B)}{f(A)}, \frac{f(\neg A,B)}{f(B)}\right)$$

Where $s_{AND}(A,B)$ is a score for co-occurrence and $s_{EX}(A,B)$ is a score for mutual exclusiveness of interactions A and B. Hereby, $f(A,\neg B)$ denotes the relative frequency of networks that contain interaction A and miss interaction B. The relative frequencies as well as scores have to exceed respective cutoffs to consider a pair of interactions AND or EX related. Scores are in the range $[0,1]$ and if the score cutoff is > 0.5, either co-occurrence-score or mutual-exclusiveness-score can exceed the cutoff, but not both.

We define characteristic interaction sets as sets of AND related interactions. Individual AND related pairs of interactions constitute the initial characteristic sets. These two-element sets are then merged to characteristic sets of higher cardinality. Two characteristic interaction sets C_x and C_y are merged if there is an AND relation between any $A \in C_x$ and $B \in C_y$, and if there is no EX relation between any $A \in C_x$ and $B \in C_y$. If there is an EX relation between any $A \in C_x$ and $B \in C_y$, then the characteristic sets C_x and C_y are considered competing, i.e. one of these characteristic sets can be present in a predicted network, but not both. Although it is possible that two characteristic sets have AND as well as EX relations between them, it was never observed during evaluations. Such rare conflicting cases need to be resolved manually.

Step 3: Groups of Networks All networks are then grouped according to the combination of characteristic sets they contain. E.g. if three characteristic sets C_x, C_y and C_z have been identified, where C_x and C_y are competing, then there are five possible combinations allowed in predicted networks (only C_x, only C_y, only C_z, C_x and C_z, C_y and C_z). If no characteristic set is present, a network is not considered for subsequent ensemble creation.

Ensemble voting can now be applied separately to each group of networks. Therefore, all interactions classified as low confident in step 1 that are part of the constituting characteristic sets are per construction enriched in the group-ensemble. Notice that each group-ensemble not only contains interactions from characteristic sets, but also all interactions previously classified as high confident, as well as other low confidence interactions.

8.3 Evaluation Results

Three hundred random gene regulatory networks were created for each of several different experimental settings (**Table 8.1**). The different experimental settings reflect different network sizes and an increasing amount of experimental data. Network sizes range from very small 5 gene networks to medium-sized networks of 15 genes. For each network a wild-type time series and the effects of a varying number of single and double knock-out perturbations were simulated:

Low amount of data Single knock-outs of about half of the genes. Affected genes were randomly selected. No double knock-outs.

Medium amount of data Single knock-outs of all genes. No double knock-outs.

High amount of data Single knock-outs of all genes. A fraction of possible double knock-outs. Affected genes were randomly selected.

Effector-target relations in the references were considered to be either activating or inhibiting with equal probability and were assigned using a given in-degree distribution of $(p_{ind}(1) = 0.7, p_{ind}(2) = 0.2, p_{ind}(3) = 0.1)$, i.e. all genes had between one and three effectors.

For each reference, 1000 network predictions were created and the 20 % with smallest root mean square deviation (RMSD) to the simulated data were used for characteristic set extraction. The following cutoffs were applied: interaction and non-interaction frequency cutoffs 0.8 and 0.1, joint frequency cutoff 0.1, AND and EX relation score cutoff 0.7. Signed voting as defined in [128] was applied to create all ensembles. Using these cutoffs, characteristic interaction sets were found in the predicted networks of a varying fraction of references, depending on network size and available data (**Table 8.1**). An actual example for reverse-engineered networks comprising a mixture of topologies is given in **Figure 8.4**.

The similarity of an ensemble, i.e. a set of weighted interactions, to a reference network can be quantified by calculating the area under the precision-recall-curve (AUPRC). Hereby, interactions are sorted according to their frequency, precision and recall with respect to the reference are calculated for all frequency cutoffs, and finally the area under the precision-recall-curve is derived. AUPRCs range between 1 (all reference interactions are top-ranked in the ensemble) and 0 (no reference interaction is present in the ensemble), thus they indicate the predictive quality of an ensemble.

The quality of predicted models and the amount of characteristic sets depend on the network size and the amounts of available experimental data. In general, we observe that prediction quality increases with an increasing amount of data, and decreases with increasing network size. The quality of predicted models in low-data experiments is quite bad with ensemble AUPRCs around 0.5 and seldom a case where the reference structure was found. Additionally, characteristic sets were hardly found in medium-sized networks. This indicates a high variability in predicted networks. The available experimental data is obviously insufficient to restrict the search space well enough to cause an enrichment of groups of networks with similar substructures. In general, the frequency of detected characteristic sets depends on network size. The larger the networks, the

more infrequent could characteristic sets be detected. We observe that the frequency of characteristic sets detected in small 5 gene networks decreases with increasing amount of data. This indicates that the increasing amount of available experimental data sufficiently restricts the search space to a single optimal area.

A	B	C	D	E	F	G
5	3	0	1.3 %	0.51/0.20	40 %	75/39/7/0
7	4	0	0.6 %	0.54/0.20	23 %	51/16/2/0
10	5	0	0.0 %	0.47/0.13	7 %	19/03/0/0
15	8	0	0.0 %	0.49/0.16	1 %	03/00/0/0
5	5	0	6.0 %	0.64/0.21	30 %	60/23/5/3
7	7	0	1.0 %	0.63/0.16	34 %	74/22/5/1
10	10	0	0.0 %	0.66/0.14	25 %	59/12/4/0
15	15	0	0.0 %	0.66/0.12	13 %	34/05/0/0
5	5	3	12.0 %	0.73/0.21	21 %	47/16/1/0
7	7	4	3.7 %	0.70/0.21	34 %	68/30/4/1
10	10	5	0.3 %	0.69/0.15	25 %	58/14/3/0
15	15	8	0.0 %	0.66/0.13	14 %	36/06/0/0

Table 8.1 Performance of reverse-engineering for varying network sizes and varying amounts of experimental data. For each of the twelve combinations of size and experimental setting, 300 random reference networks were created. For each reference, a wild-type time series and a varying number of knock-out perturbations were simulated. Knocked-out genes were randomly chosen. A) Number of genes in networks. B) Number of different simulated single knock-out experiments. C) Number of different simulated double knock-out experiments. D) Percentage of cases where a predicted network was identical to the reference. E) AUPRC of the ensemble of all networks (mean/standard deviation). F) Percentage of cases where characteristic interaction sets have been identified. G) Number of cases where 2/3/4/5 characteristic sets were identified.

Networks are grouped according to the presence or absence of sets of related low confident interactions. Thus, these interactions should be either frequently present or absent within the resulting groups of networks. Other variable low confident interactions should be not affected. To evaluate this, the entropies of group-ensembles were calculated and compared to the entropy of the ensemble of all networks. An ensemble's entropy can be used as a measure of its overall confidence, i.e. ensembles with a large proportion of low confidence interactions (intermediate frequencies) have a higher total entropy compared to ensembles with many high confidence interactions (very high or low frequencies). An ensemble's entropy is defined as $-\sum_A f(A) \cdot \log_2(f(A))$, where $f(A)$ is the relative frequency of interaction A. Group-ensemble entropies are on average reduced to 45 % compared to the entropy of the ensemble of all networks (**Figure 8.5A**).

Each created group of networks constitutes an alternative hypothesis about the true gene regulatory network, and it is essential to decide which of those is most similar to the reference. Typically, one would assume that the hypothesis which is superior in explaining experimental data is most similar to the reference. But only in 31.2 % of the cases where two or more group-

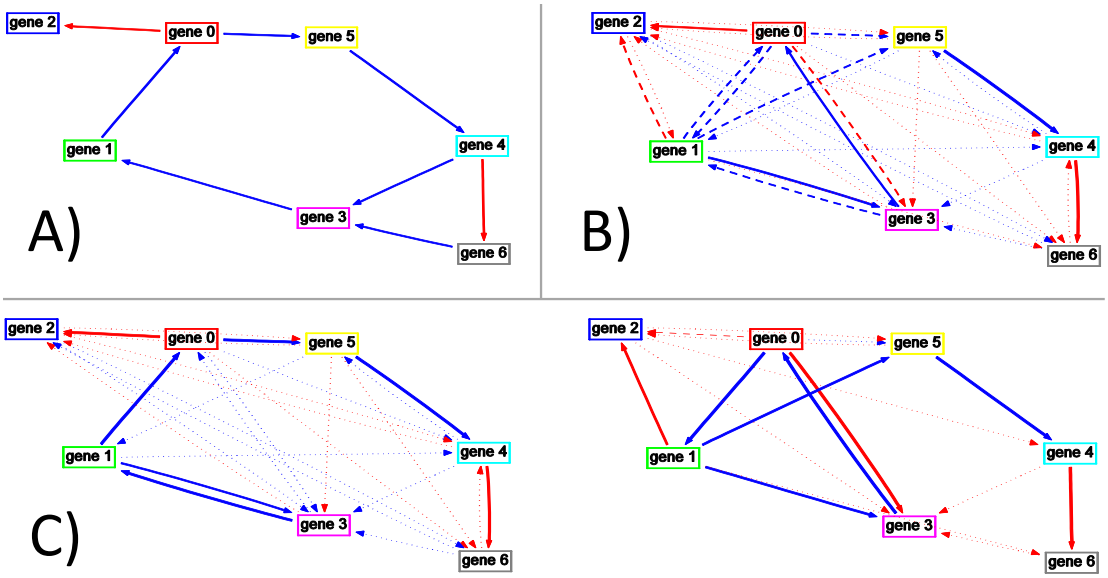


Figure 8.4 Example for extracted group-ensembles. Using simulated data from a random reference network (A), the applied reverse-engineering algorithm created a set of networks which were combined to an ensemble (B). Two group-ensembles were derived using the described characteristic interaction set extraction approach (C). Both group-ensembles explain the simulated data very well (average RMSD 0.075 and 0.081) but effector-target relations differ strongly (AUPRC to reference 0.898 and 0.311).

ensembles were found, the group-ensemble with highest similarity to the reference also has the smallest root mean square deviation to the simulated data. Hereby, the group-ensembles RMSD was calculated by averaging the RMSD of all contained networks. Additionally, the RMSD distribution of ensembles with highest similarity to the reference is not significantly different to the RMSD distribution of all other ensembles (Wilcoxon rank sum test, p-value ≈ 0.61). Thus, the score of a group-ensemble, i.e. the ability to reproduce the experimental data by the contained networks, is not suited to decide for one of these hypotheses.

We checked whether some group-ensembles have an increased AUPRC as compared to the ensemble of all networks and thus are better predictions of the reference network than the ensemble of all networks (**Figure 8.5B**). A positive correlation between AUPRCs of group-ensembles and the precision of characteristic sets was observed. The precision of a characteristic set is the fraction of its interactions that could be found in the reference. This correlation can be expected, as characteristic interactions are per construction enriched in group-ensembles, and only if these characteristic interactions could also be found in the reference, then the AUPRC should increase due to this enrichment.

The networks which are most similar to the reference could be identified by further experimental evidence or additional prior knowledge concerning the presence of individual interactions. If the presence (or absence) of a certain interaction could be established, all hypotheses lacking (or containing) this interaction could be rejected. Interactions with intermediate frequencies are well suited targets for validation, as knowledge about their presence or absence would allow to reject a substantial proportion of networks by few experiments. Therefore, we simu-

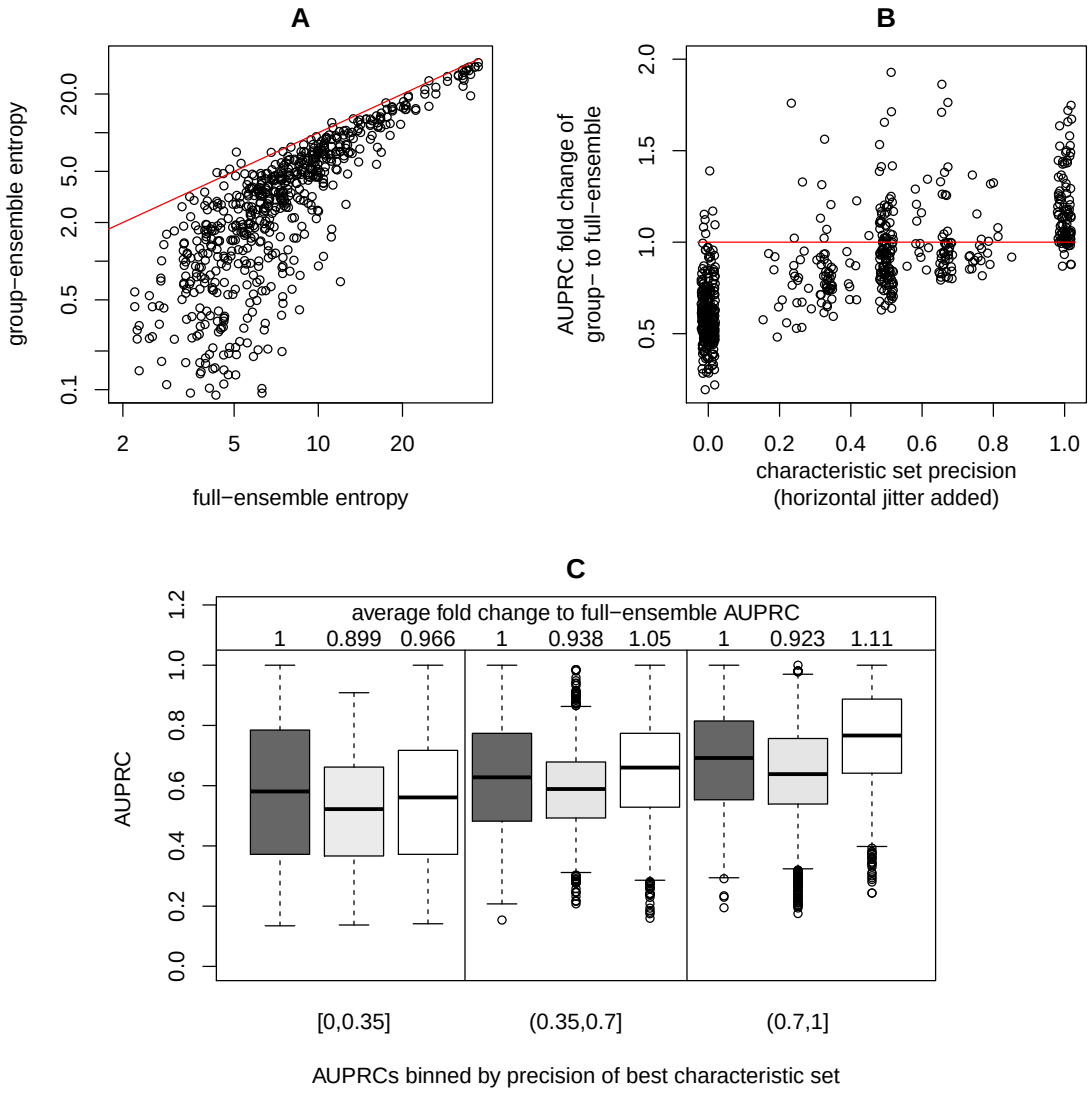


Figure 8.5 Entropy and AUPRC evaluation results. (A) The entropy of group-ensembles is on average decreased to 45 % as compared to the entropy of the ensemble of all networks (full-ensemble). This is caused by the reduced fraction of low confidence interactions. (B) AUPRCs of group-ensembles are increased if their characteristic sets are present in the reference. The precision of characteristic sets ranges between 1 (all interactions are present in the reference) and 0 (no interaction is present in the reference). A small amount of horizontal jitter (<0.02) was added to the precision values for better visualization. The red lines indicate identity. (C) Rejecting alternative hypothesis by testing for the presence of characteristic set interactions (white box-plots) in general increases AUPRC, while testing for other low confidence interactions (gray box-plots) has a less pronounced or even negative effect. Thus, interactions that are predicted to be co-occurring with other interactions are preferred targets of further experimental verification. The full-ensemble AUPRC distribution is shown as dark-gray box-plot.

lated such experimental validations of individual low confidence interactions to test whether co-occurring interactions are especially suited for it. For each low confidence interaction, we checked whether it was actually present in the reference and accordingly rejected all networks containing (or lacking) the tested interaction. On average, the AUPRC of the ensemble of remaining networks increased only when testing for characteristic set interactions (**Figure 8.5C**). Thus, it is beneficial for experimental designs to distinguish between characteristic set interactions and other highly variable interactions beforehand.

8.4 Discussion

Knowing the mutual dependencies between interactions and grouping networks respectively has some clear-cut benefits. Firstly and obviously, these dependencies between interactions become accessible for interpretation. Second, by grouping the typically hundreds or thousands of high scoring networks according to local similarities, the results of a reverse-engineering run become more interpretable. If ensemble voting is applied to each group, the fraction of low confidence interactions within each group-ensemble is decreased in favor of high confidence interactions. Third, guidance for the design of further experiments is provided. As either all interactions of a characteristic set are present or all interactions of the competing one, experimental verification of a single interaction should be sufficient to identify which characteristic set is actually realized in the biological system.

The described approach refines ensembles of gene regulatory networks as predicted by dynamic model based reverse-engineering algorithms. Thus, it depends on their prediction quality, proper sampling of models and the availability of a sufficiently large number of networks (a least a few hundred). These networks could be created by non-deterministic optimization, by separate (deterministic) optimizations based on different experimental data or cross-validation [129], by incorporation of a varying amount of prior knowledge, or a combination of these.

The characteristic set extraction algorithm is intended to be simple, comprehensible, traceable and computationally fast. Pairs of co-occurring (AND related) and mutually exclusive (EX related) interactions are identified by calculating scores from their joint and individual frequencies. Notice that the typical number of observed effector-target relations is much less than the number of possible pairs, thus counting joint occurrences can be done very fast. Our definition of a score of AND relations was derived from the concept of confidence used in association rule learning [130]. Hereby, the association rules $(A \Rightarrow B)$ and $(B \Rightarrow A)$ were combined by the *minimum* conjunction. The score of EX relations was defined analogously by combining $(\neg A \Rightarrow B)$ and $(\neg B \Rightarrow A)$. The interpretation of pairwise relations is straightforward. Both AND related interactions have to be present to achieve a biologically meaningful effect. EX related interactions might either represent redundancies, which are discouraged as sparseness of networks is typically demanded in reverse-engineering, or the presence of both interactions might contradict the reference data. Pairs of co-occurring interactions are then merged to (competing) characteristic sets. This resembles a bottom-up procedure where large entities are constructed by joining of frequent smaller parts, and can be done very efficiently.

Cutoffs applied during characteristic set extraction are used to distinguish reliable observati-

ons (here, occurrence of certain interactions or characteristic interaction sets) from spurious ones, which might be artifacts of the applied reverse-engineering approach (noise). The specific choice of cutoffs is interdependent with the number and frequency of distinct characteristic sets hidden in the predictions. Additionally, the number of high-scoring networks with different topologies, and therefore the number of characteristic sets, depends on the reference topology, available data, and the applied mathematical framework. Therefore, cutoffs have to be assessed case-specific, e.g. by starting with relatively stringent cutoffs, repeatedly reducing them and assessing extracted characteristic sets by number, size and biological meaning.

The presented procedure can be extended in various ways, e.g automatic adjustment of cutoffs, considering scores of predicted networks during ensemble voting to give high-scoring networks a higher weight, fuzzy assignment of interactions to characteristic sets and fuzzy grouping of networks to account for random variations, and using grouped networks as priors for repeated rounds of reverse-engineering and characteristic set extraction to explore different network hypotheses in more detail.

Author's contribution The content of Section 8 has been compiled to a manuscript and submitted for publication. Co-authors are Robert Küffner who developed the genetic algorithm used to create network predictions, Jonas Zierer who participated as student assistant in the implementation of the described procedure, and Ralf Zimmer who supervised to the work. The author developed the characteristic set extraction algorithm and wrote the manuscript.

Summary and Outlook

Chapter 9

Summary and Outlook

The two main parts of this work introduced the Petri net and fuzzy logic (PNFL) modeling framework and associated reverse-engineering approaches. The main aspects and findings of these parts are shortly summarized in the following.

9.1 Petri Net and Fuzzy Logic Based Modeling

We have shown that PNFL models can represent states of biological entities and their interactions, can mimic mass-action kinetics, and approximate Hill kinetics (Sections 2 and 3). PNFL models are capable of reproducing the complex behavior of biological systems and can be analyzed using standard analysis techniques (Section 4). They can be used to predict a system's behavior based on varying experimental conditions and are therefore suited for hypotheses testing (Section 5). The main advantage of PNFL models is the pronounced inherent graphical and structured representation of not only network connectivity, but also of the functional aspects of the model: the fuzzy sets and rule bases of fuzzy logic systems. These aspects strongly facilitate model creation and understanding. A representation based on fuzzy sets inherently includes an interpretation of the described states with respect to a biologically relevant aspect, for example with respect to functional implications, typical values, etc. Fuzzy logic systems have a one-to-one correspondence to linguistic descriptions of the represented processes. Linguistic descriptions of interactions between entities can be converted into the rule bases and, *vice versa*, each rule base is a linguistic description of the represented biological process. Thus, the Petri net and fuzzy logic modeling framework allows for a straightforward conversion of qualitative knowledge and descriptions into executable models.

9.2 Reverse-Engineering

PNFL models can be reverse-engineered based on experimental measurements using the genetic algorithm described in Section 6. The reverse-engineered models were some of the best predictions in the DREAM4 in-silico network reconstruction challenge. This further supports that PNFL models are well suited to model reference systems, and that the genetic algorithm is very

well suited for reverse-engineering PNFL models. Unfortunately, the genetic algorithm performs poorly on larger networks due to the massive increase of the search space, and a single prediction has a runtime of several hours. Although the number of possible models is already strongly restricted due the restricted definition of valid PNFL models, including a reduced selection of fuzzy logic systems and generalized mean parameters, the search space grows exponentially with an increasing number of entities. Section 7 describes an iterative procedure that allows for the reverse-engineering of larger networks. This iterative procedure results in an increase of prediction performance for models of up to 120 genes. A problem of the iterative procedure seems to be that the genetic algorithm gets stuck in local optima after the first iteration and therefore model sizes hardly increase although the candidate interaction sets are relaxed. The post-processing of ensemble predictions, as described in Section 8, further improves the information gain from network predictions. It is suited to distinguish different classes of models that result from the presence of multiple, similarly scoring optima in the search space. Such a post-processing is meaningful if there is a moderate amount of variability in predicted models. As especially in large networks many combinations of characteristic interaction sets are possible, an analysis of subnetworks might become necessary.

9.3 Outlook

Structural analysis using Petri net analysis techniques There are several techniques that allow structure-based analysis of Petri nets, the most prominent of these are based on place and transition invariants. Unfortunately, p- and t-invariants can not be computed straightforwardly for PNFL models. The reason for this is that an incidence matrix of the Petri net is required for the standard p- and t-invariant analysis. To derive such an incidence matrix, fixed arc-weights are required, i.e. constant functions have to be assigned to arcs. This is obviously not given for PNFL models, as fuzzy logic systems and most other allowed functions are not constant but depend on input states. One approach to solve this problem could be to convert PNFL models into discrete logic models and further into standard Petri nets. A t-invariant analysis of these Petri nets could be used to draw conclusions about potential steady states or oscillating states of the discrete logic model. Whether the achieved results can be applied to the original PNFL model remains to be investigated.

Further improving reverse-engineering of large networks The data-driven scores used to restrict the search space are very basic. One could try to apply more involved prediction approaches like ARACNE [48], MRNET [49], etc. to derive initial static models of the system, which could then be converted into dynamic PNFL models. These can subsequently be extended by adding interactions that are necessary for the expected dynamic behavior, or reduced by removing redundant interactions. Nevertheless, the problem of search space explosion persists. We presume that very large networks can only then be predicted with success, if networks can be partitioned into subnetworks that in turn can be predicted independently in a modular fashion. These subnetworks could then be integrated into a full network.

9.4 Conclusion

The ambition of systems biology is to get a comprehensive understanding of the dynamics of biological systems. Especially due to the complexity and size of networks, the various levels of regulations, and the individual or cell-type specific variations, detailed quantitative information about entities and interactions is often missing and cumbersome to obtain. This includes for example reliable and fine grained concentration time course measurements or measurements of reaction rates. However, the PNFL modeling technique can be readily applied to such problems. It helps to formulate (initial) hypotheses as executable, and easy to interpret, dynamic models, even if detailed kinetic information is not yet available. As for any other modeling technique as well, data interpretation, pre-processing and selection are still necessary work steps preceding model creation. According to the available data and the problem statement at hand, fuzzy sets, fuzzy concepts and rule bases have to be carefully designed, both for manual model creation as well as for automated reverse-engineering.

The incorporation of prior information of various types and from various sources is getting increasingly important due to the fast development of high-throughput experimental techniques. In most cases, such prior information is qualitative. Especially the ENCODE project [131] created a vast source of qualitative experimental information that can be used to supplement the creation of (genome-wide) gene regulatory networks: knowledge about the presence and localization of proximal and distal transcription factor binding sites allows to get an initial idea about effector-target relationships, activating or inhibiting roles of transcription factors, or mutually dependencies between them; knowledge about chromatin conformation and DNA methylation patterns allows to deduce cell-type specific transcriptional activity; knowledge about the various non-coding RNAs even allows to include another level of regulation, etc. Such qualitative information can and should be used to improve dynamic models and to guide reverse-engineering. Especially concerning the incorporation of qualitative prior information, we see a great potential for the application of the PNFL modeling approach.

Due to the easily interpretable descriptions of entities and interactions by fuzzy sets and natural language based rules, the properties and functionalities of PNFL models become apparent. Therefore, PNFL models are especially comprehensible. This facilitates the creation and understanding of dynamic models. We see Petri net and fuzzy logic based modeling as a valuable addition to the pool of established modeling techniques, suited to the challenges of incorporating heterogeneous and not easy to quantify data.

Appendix

Appendix A

Ordinary Logic Reasoning and Fuzzy Logic - Theoretical Background

This theoretical foundations justify the rule bases of fuzzy logic systems which are used in our PNFL framework. The content of this section was compiled based on [107, 132, 133]. Before introducing fuzzy sets, we will first go through some aspects of ordinary logic reasoning which will then be generalized to fuzzy logic reasoning.

A.1 Ordinary Logic Reasoning

We want to use rule bases to define the dynamics of a system, i.e. to derive the new state of an entity from a given set of states of other entities. Those rule bases are given as linguistic if-then sentences. To implement such sentences in a computational framework, they have to be converted to (mathematical) functions first. These functions have to map several specified input sets to a single, specified output set. In the following section we will show how such functions can be derived for ordinary sets using ordinary relations and logic implication and that the conclusions are backed by the modus ponens.

A.1.1 Modus Ponens

Definition 1 *One of the most important inference rules in traditional propositional logic is the modus ponens*

Premise 1 (Fact)	*x is A*
Premise 2 (Rule)	*if x is A then y is B*
Conclusion	*y is B*

which allows to derive a conclusion from given (combined) propositions.

If both premises are true, i.e. the fact is true ("x is actually A") and the rule is valid, then it can be logically concluded that the conclusion must also be true (the argument is *sound*). If one or

both of the premises are false, the modus ponens does not apply. We want to emphasize that false premises *do not* imply that the conclusion has to be false too. Modus ponens does not allow *any* deductions from false premises.

A.1.2 Ordinary Relations

Definition 2 *A (two-valued) relation $R \subseteq X \times Y$ contains pairs (x,y) which are in some kind of relation described by R, i.e. (x,y) fulfill conditions defining the relation.*

E.g. let $G = \{g_1, g_2, g_3\}$ be a set of gene products and $C = \{c_1, c_2\}$ be a set of cell compartments. Then a relation $R \subseteq G \times C$ may describe that certain gene products can be found in certain cell compartments only: $R = \{(g_1, c_1), (g_2, c_2), (g_3, c_1), (g_3, c_2)\}$ where R contains an element (g_i, c_j) if and only if gene product g_i can be found in compartment c_j.

Definition 3 *Let $R \subseteq X \times Y$ be a relation, then the image R_M of the relation with respect to the set $M \subseteq X$ is defined as*

$$R_M = \left\{ y \in Y \;\middle|\; \exists x \in X : \; (x,y) \in R \wedge x \in M \right\} \tag{A.1}$$

and contains all elements $y \in Y$ which are in relation to an element $x \in M$.

We want to point out that a set as well as a relation and its image can be represented by indicator functions

$$\begin{aligned}
I_M : M \to \{0,1\}, \qquad x \quad &\mapsto \begin{cases} 1 & if\ x \in M \\ 0 & else. \end{cases} \\
I_R : X \times Y \to \{0,1\}, \qquad (x,y) \quad &\mapsto \begin{cases} 1 & if\ (x,y) \in R \\ 0 & else. \end{cases} \\
I_{R_M} : Y \to \{0,1\}, \qquad y \quad &\mapsto \begin{cases} 1 & if\ \exists x \in X : \; (x,y) \in R \wedge \; x \in M \\ 0 & else. \end{cases}
\end{aligned} \tag{A.2}$$

An equivalent definition of R_M can be given by using indicator functions instead of logical notations

$$I_{R_M}(y) = \sup \left\{ I_M(x) \cdot I_R(x,y) \;\middle|\; x \in X \right\} \tag{A.3}$$

which yields $I_{R_M}(y) = 1$ if and only if there is an $x \in M$ which is in relation R to the given $y \in Y$ and zero otherwise. The supremum (sup) of a set of real numbers is the smallest real number that is greater or equal to every number in this set. In contrast to the greatest element of a set, the supremum is not necessarily part of the set.

A.1.3 Logic Implication

An implication $x \in A \to y \in B$ for $A \subseteq X$ and $B \subseteq Y$ can be defined using the following relation:

$$R_{A \to B} = \left\{ (x,y) \in X \times Y \;\middle|\; x \in A \to y \in B \right\} = (A \times B) \cup \bar{A} \times Y \tag{A.4}$$

The image with respect to the set A is $R_{A\to B}[A]$ and its indicator function is defined as

$$I_{R_{A\to B}[A]} : Y \to \{0,1\}, \quad y \mapsto \begin{cases} 1 & if\ \exists x \in A\ and\ (x \in A \to y \in B) \\ 0 & else. \end{cases} \tag{A.5}$$

or equivalently

$$I_{R_{A\to B}[A]} : Y \to \{0,1\}, \quad y \mapsto \sup\left\{ I_A(x) \cdot I_{R_{A\to B}}(x,y) \;\middle|\; x \in X \right\}. \tag{A.6}$$

The definition of the indicator function has the structure of a modus ponens with first premise (fact) $x \in A$, second premise (rule) *if* $x \in A$ *then* $y \in B$ and the conclusion $y \in B$. Obviously, $I_{R_{A\to B}[A]}$ indicates the set of all sound conclusions deduced by modus ponens. Therefore, it can be seen as a mapping of the input set A to the output set $R_{A\to B}[A] \subseteq B$ with respect to the given implication rule. This functions allows a general statement about conclusions of the rule. It is suited to answer the question, which conclusions are possible, given an unspecified element of X as premise (i.e. independent from a specific premise $x' \in X$).

We need to answer the question, which conclusion can be derived from a specific input $x' \in X$. This is easily done by restricting the image to the singleton set $\{x'\}$:

$$\begin{aligned} I_{R_{A\to B}[\{x'\}]} : Y \to \{0,1\}, \qquad y &\mapsto \sup\left\{ I_{[\{x'\}]}(x) \cdot I_{R_{A\to B}}(x,y) \;\middle|\; x \in X \right\} \\ \Leftrightarrow \quad y &\mapsto I_{R_{A\to B}}(x',y) \end{aligned} \tag{A.7}$$

The resulting set $R_{A\to B}[\{x'\}]$ then contains all valid conclusions given a specific x'. Obviously this conclusion is still backed by the modus ponens if $x' \in A$, i.e. the argument is sound. But unfortunately, a serious problem arises when applying this type of inference to any $x' \notin A$: the implication operation evaluates to truth whenever a false antecedent is given (independent of the consequent!). Therefore $I_{R_{A\to B}[\{x'\}]}$ will always yield 1 (true) if $x' \notin A$ is chosen. In this case the argument is not sound, as the modus ponens does not apply. Although the deduction is still (logically) correct, it violates common sense and is not desired in any practical application. It violates the cause and effect assumption, as here *non-cause leads to anything*. We will point out a practical solution to this problem in Section A.3 when considering fuzzy implication.

In the next sections we will introduce a more formal definition of fuzzy sets, define operations on fuzzy sets and extend the relations and implications introduced above to finally define the rule bases working on fuzzy sets.

A.2 Fuzzy sets

Definition 4 *A fuzzy set μ_M defined over a domain of discourse D is a function $\mu_M : D \to [0,1]$ which assigns to each element $x \in D$ a degree of membership $\mu_M(x)$ to the fuzzy set μ_M. The set of all fuzzy sets over D is denominated $F(D)$.*

If $\mu_M(x) \in \{0,1\}$ for all $x \in D$, then μ is the indicator function of a classical, or crisp, set $M \subseteq X$.

Definition 5 *The process of mapping a crisp point x to* $[0,1]$ *using a fuzzy set* μ_M *is called fuzzification and* μ_M *can be called a fuzzifier.*

In most cases the domain of discourse is the set of real numbers $D = \mathbb{R}$ or any interval on $\mathbb{R}$. Then μ is a real-valued function and can be displayed as for example shown in Section 2. Typically, fuzzy sets describe linguistic terms like "nearly inactive", "at an average level" or "close to 100" and describe either an imprecise value or interval. Such fuzzy sets should be *convex*, i.e. they should monotonically increase(decrease) when converging(diverging) to(from) a specific value or a specific interval, e.g. when "getting closer to 100" or "leaving the average level". The most commonly used convex functions for fuzzy sets are triangular, trapezoidal or Gaussian functions. The choice of number, shape and parameters of fuzzy sets (e.g. median and standard deviation of a Gaussian) define the context and user dependent setup of the fuzzy model.

One of the greatest advantages of a discretization of a state space (domain of discourse) using fuzzy sets compared to the use of crisp sets is the fact that it is meaningful to overlap fuzzy sets. This allows us to express that e.g. the expression of a gene is "somewhere between its normal level and its over-expressed state" instead of having to define it either as over-expressed or not (as both at the same time would be contradictory).

A.2.1 Operations on Fuzzy Sets

T-Norms and T-Conorms The classical logic assumes that propositions are either true or false, but not both true and false. In fuzzy logic there is a gradual transition from truth to falsehood. Therefore, functions like the conjunction and disjunction have to be extended to operate on the unity interval $w_\wedge, w_\vee : [0,1]^2 \to [0,1]$ and the most basic requirement for such "fuzzy" conjunction and disjunction functions is to yield the same results as their classical counterparts when provided with either 0 or 1. Candidates for such conjunction and disjunction functions are T-norms (triangular norms, denoted $t(\dots)$ or $\star$) and T-conorms (denoted $s(\dots)$ or $\circ$), respectively. The most commonly used T-norms are

minimum	$t(\alpha,\beta) = min\{\alpha,\beta\}$
bounded product	$t(\alpha,\beta) = max\{\alpha+\beta-1,0\}$
algebraic product	$t(\alpha,\beta) = \alpha\cdot\beta$

Examples for commonly used T-conorms are

maximum	$s(\alpha,\beta) = max(\alpha,\beta)$
bounded sum	$s(\alpha,\beta) = min\{\alpha+\beta,1\}$
algebraic sum	$s(\alpha,\beta) = \alpha+\beta-\alpha\cdot\beta$

All T-norms and T-conorms are commutative, associative and monotone, while additionally T-norms fulfill $t(\alpha,1) = \alpha$ and T-conorms fulfill $s(\alpha,0) = \alpha$ (**Figure A.1**).

Union, Intersection and Complement of Fuzzy Sets The intersection of fuzzy sets can be derived from the respective definition for classical sets. An element x is in the intersection of two classical sets A and B if it is member of both sets:

$$x \in A \cap B \;\Leftrightarrow\; x \in A \;\wedge\; x \in B \tag{A.8}$$

The intersection of two fuzzy sets μ_A and μ_B with respect to a T-norm t can then be defined as fuzzy set $\mu_{A \cap_t B}$ with

$$\mu_{A \cap_t B}(x) \;=\; t(\mu_A(x), \mu_B(x)) \;=\; \mu_A(x) \star \mu_B(x) \tag{A.9}$$

Analogously, an element x can be in the union of two classical sets A and B:

$$x \in A \cup B \;\Leftrightarrow\; x \in A \;\vee\; x \in B \tag{A.10}$$

And the union of two fuzzy sets μ_A and μ_B with respect to a T-conorm s can be defined as fuzzy set $\mu_{A \cup_s B}$ with

$$\mu_{A \cup_s B}(x) \;=\; s(\mu_A(x), \mu_B(x)) \;=\; \mu_A(x) \circ \mu_B(x) \tag{A.11}$$

The complement of a fuzzy set is derived from the definition $x \in \bar{A} \;\Leftrightarrow\; \neg(x \in A)$ for classical sets and corresponds to the fuzzy set $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$ (**Figure A.1**).

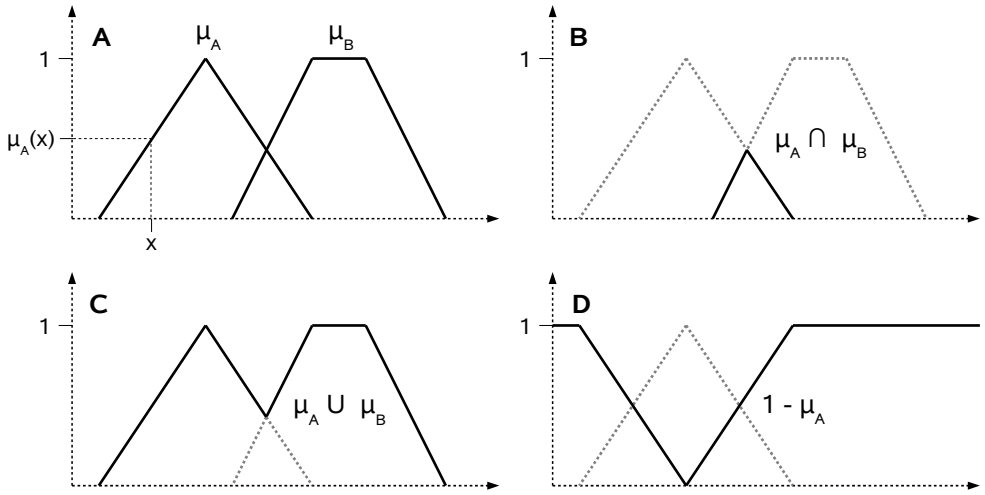


Figure A.1 Operations on fuzzy sets. A) Two fuzzy sets μ_A and μ_B are defined with triangular and trapezoidal shape, respectively. A (crisp) point x is fuzzified by μ_A, i.e. $\mu_A(x)$ is calculated using the triangular function. **B)** The intersection $\mu_{A \cap_t B}(x)$ using the minimum T-norm. **C)** The union $\mu_{A \cup_s B}(x)$ using the maximum T-conorm. **D)** The complement of μ_A.

A.3 Fuzzy Logic Reasoning

The use of fuzzy sets in logic reasoning leads to an extension of the classical modus ponens. This extension facilitates to reason with gradual truth values, i.e. to infer conclusions from vague, imprecise knowledge.

A.3.1 Generalized Modus Ponens

Definition 6 *In fuzzy logic the modus ponens is extended to the generalized modus ponens*

Premise 1 (Fact)	*x is* A^*
Premise 2 (Rule)	*if x is A then y is B*
Conclusion	*y is* B^*

The *generalized modus ponens* applies as long as there is a non-zero degree of similarity between the fuzzy set A^* of premise 1 (fact) and the fuzzy set A of the antecedent of premise 2 (rule), and as long as there is a non-zero similarity between the fuzzy set B of the consequent of premise 2 and the fuzzy set B^* of the conclusion. If additionally the degrees of truth of the premises are non-zero, a non-zero degree of truth for the conclusion can be deduced. In general, it holds that the higher the degree of truth of the premises, the higher the degree of truth of the conclusion is. When A, A^*, B, B^* are considered to be ordinary sets with $A = A^*$ and $B = B^*$ the generalized modus ponens reduces to the modus ponens.

A.3.2 Fuzzy Relations and Implication

A fuzzy set is a generalized ordinary set and analogously a fuzzy relation can be seen as a generalized ordinary relation. A two-valued fuzzy relation μ_R assigns a degree of membership to each pair $(x,y) \in D_x \times D_y$ reflecting the strength of relation between x and y. Obviously, such a fuzzy relation can be defined as a fuzzy set with domain of discourse $X \times Y$.

The image of a fuzzy relation $\mu_R : D_x \times D_y \rightarrow [0,1]$ with respect to a fuzzy set $\mu_M : D_x \rightarrow [0,1]$ is defined as a fuzzy set

$$\mu_{R[\mu_M]}(y) = \sup \left\{ \mu_M(x) \star \mu_R(x,y) \;\middle|\; x \in D_x \right\} \tag{A.12}$$

with D_y as domain of discourse and $\star$ as an intersection of fuzzy sets using a T-norm. The image of an ordinary relation (Equation A.3) is a special case of this fuzzy set with the product as T-norm.

Implication The fuzzy implication relation $\mu_{A \rightarrow B}(x,y)$ measures the degree of truth of the implication $x \in A \rightarrow y \in B$ and the image of this fuzzy implication relation with respect to the (premise) fuzzy set μ_{A^*} is

$$\mu_{B^*}(y) = \sup \left\{ \mu_{A^*}(x) \star \mu_{A \rightarrow B}(x,y) \;\middle|\; x \in D_x \right\} \tag{A.13}$$

which has the structure of a generalized modus ponens. The right-hand-side of this equation defines the degree of truth $\mu_{B^*}(y)$ of a conclusion $y \in D_y$ depending on an uncertain fact $\mu_{A^*}(x)$ and an uncertain rule $\mu_{A \rightarrow B}(x,y)$. As mentioned in Section A.1.3, we want to derive a conclusion from a specific $x' \in D_x$. In ordinary logic, this is done by restricting the image to the set $\{x'\}$. In fuzzy logic this restriction is performed by specifically defining μ_{A^*} as a fuzzifier for x', for example $\mu_{A^*}(x) = 1$ for $x = x'$ and $\mu_{A^*}(x) = 0$ otherwise. Such a μ_{A^*} is called a *singleton fuzzifier*.

In general the fact μ_{A^*} can be used to reflect our uncertainty about a crisp point $x' \in D_x$, which could for example be a measurement of a parameter of any type (e.g. a concentration). Therefore one could choose μ_{A^*} to have its maximum at the observed value x' and could choose a wider support if the uncertainty about x' is large and a small support otherwise. The choice of μ_{A^*} influences the interval containing a possible supremum of Equation A.13. Notice that the supremum can be different from the actual parameter x'.

Singleton fuzzification is widely used as it leads to a tremendous reduction in computational cost due to the disappearance of the supremum operation:

$$\begin{aligned} \mu_{B^*}(y) &= \mu_{A^*}(x') \star \mu_{A\to B}(x', y) \\ &= 1 \star \mu_{A\to B}(x', y) \\ &= \mu_{A\to B}(x', y) \end{aligned} \tag{A.14}$$

So in the case of singleton fuzzification the degree of truth of y is deduced by an uncertain rule from a (certain) measurement x'.

To evaluate Equation A.13 a meaningful membership function for $\mu_{A\to B}(x,y)$ has to be chosen. A possible candidate could be derived from the logical equivalence $(p \to q) \Leftrightarrow (\neg p \vee q)$ by using a T-conorm:

$$\mu_{A\to B}(x,y) = \mu_{\bar{A}\cup_s B}(x,y) = s(1-\mu_A(x),\ \mu_B(y)) \tag{A.15}$$

But this choice would cause the same problem as mentioned in Section A.1.3, i.e. yielding non-zero results although the antecedent $\mu_A(x)$ is zero. Diverging from the standard definition of implication in propositional logic, a T-norm evaluates to zero if any of its arguments is zero. The monotonicity of T-norms assures that the higher the degrees of truth of its arguments are, the higher the degree of truth of the result is. Thus, a T-norm can be seen as an extension of an implication operation which preserves cause and effect, and therefore it is reasonable to choose one as fuzzy implication operator. Using the two most common T-norms minimum and product, one could derive the following membership functions for the fuzzy implication:

$$\begin{aligned} \mu_{A\to B}(x,y) &= \min\{\mu_A(x),\ \mu_B(y)\} \\ \mu_{A\to B}(x,y) &= \mu_A(x)\cdot\mu_B(y) \end{aligned} \tag{A.16}$$

Until now we only considered two-valued (fuzzy) relations, which only allow implications from a single antecedent to a single consequent. The extension to multiple antecedents is straightforward by replacing the single element x by a vector of elements $\vec{x} \in D_1 \times D_2 \times \cdots \times D_n$. Equation A.13 then extends to

$$\begin{aligned} \mu_{B^*}(y) &= \sup\left\{ \mu_{\vec{A}^*}(\vec{x}) \star \mu_{\vec{A}\to B}(\vec{x}, y) \;\middle|\; \vec{x} \in D_1 \times \cdots \times D_n \right\} \\ &= \sup\left\{ \mu_{A_1^*\times\cdots\times A_n^*}(\vec{x}) \star \mu_{A_1\times\cdots\times A_n\to B}(\vec{x}, y) \;\middle|\; \vec{x} \in D_1 \times \cdots \times D_n \right\} \end{aligned} \tag{A.17}$$

where $\mu_{A_1^*\times\cdots\times A_n^*}(\vec{x})$ is the Cartesian product realized by combining $\mu_{A_1^*}(x_1), \ldots, \mu_{A_n^*}(x_n)$ by T-

norms to $\mu_{A_1^*}(x_1) \star \cdots \star \mu_{A_n^*}(x_n)$ leading to

$$\begin{aligned}\mu_{B^*}(y) &= \sup\left\{ \bigcap_{i=1}^{n}{}_t\, \mu_{A_i^*}(x_i) \star \bigcap_{i=1}^{n}{}_t\, \mu_{A_i}(x_i) \star \mu_B(y) \;\middle|\; (x_1,\ldots,x_n) \in X_1 \times \cdots \times X_n \right\} \\ &= \sup\Big\{ \mu_{A_1^*}(x_1) \star \cdots \star \mu_{A_n^*}(x_n) \star \mu_{A_1}(x_1) \star \cdots \star \mu_{A_n}(x_n) \star \mu_B(y) \\ &\qquad \Big| \; (x_1,\ldots,x_n) \in X_1 \times \cdots \times X_n \Big\} \end{aligned} \tag{A.18}$$

which in case of singleton fuzzification reduces to

$$\begin{aligned}\mu_{B^*}(y) &= \mu_{\vec{A}\to B}(\vec{x'},y) = \bigcap_{i=1}^{n}{}_t\, \mu_{A_i}(x_i') \star \mu_B(y) \\ &= \mu_{A_1}(x_1') \star \cdots \star \mu_{A_n}(x_n') \star \mu_B(y) \end{aligned} \tag{A.19}$$

Keep in mind that this image of the fuzzy implication relation $\mu_{A\to B}(\vec{x},y)$ is itself a fuzzy set with D_y as its domain of discourse. Equation A.18 or its reduced form A.19 deduce a valid conclusion from a given explicit premise $\vec{x'}$ backed by generalized modus ponens. Due to the choice of a T-norm as implication operator the problem of an unsound argument as a result of a false premise disappears.

A.3.3 Fuzzy Logic Systems

We have seen how a fuzzy implication relation can be used to map a set of antecedents to a consequent, i.e. how a conclusion can be derived from a set of premises using a single rule of the form *if* $\vec{x} \in \vec{A}$ *then* $y \in B$. In practice one typically wants to derive a single result from a *set of rules* which define consequents for different (or each) combinations of antecedents.

If a conclusion should be derived from several rules, the individual results have to be combined appropriately. The image of a set of fuzzy implication relations is the disjunction of their individual images using a T-conorm:

$$\mu_{B^*}(y) = \bigcup_{j=1}^{m}{}_s\, \mu_{B_j^*}(y) = \bigcup_{j=1}^{m}{}_s\, \sup\left\{ \mu_{\vec{A}_j{}^*}(\vec{x}) \star \mu_{\vec{A}_j\to B_j}(\vec{x},y) \;\middle|\; \vec{x} \in D_x \right\} \tag{A.20}$$

which can be further reduced when using *singleton fuzzification* to

$$\mu_{B^*}(y) = \bigcup_{j=1}^{m}{}_s\, \mu_{B_j^*}(y) = \bigcup_{j=1}^{m}{}_s\, \left(\bigcap_{i=1}^{n}{}_t\, \mu_{A_{j,i}}(x_i') \star \mu_{B_j}(y) \right) \tag{A.21}$$

Definition 7 *A fuzzy inference engine is a function* $D_1 \times \cdots \times D_n \to F(D_y)$ *which maps a vector of crisp inputs* $\vec{x} \in D_1 \times \cdots \times D_n$ *to a conclusion fuzzy set* $\mu_{B^*}(y) \in F(D_y)$*. The mapping is specified by a fuzzifier, a set of rules (fuzzy rule base)*

$$R_j : \mathit{IF}\ x_1\ \mathit{is}\ A_{j,1}\ \mathit{AND}\ x_2\ \mathit{is}\ A_{j,2}\ \mathit{and}\ \ldots\ \mathit{and}\ x_{n_j}\ \mathit{is}\ A_{j,n_j}\ \mathit{THEN}\ y\ \mathit{is}\ B_j,$$

where n_j *is the number of premises of rule* j *and* $A_{j,i}$ *specifies the fuzzy set of premise* i *in rule* j*, and the definition of a T-norm and T-conorm for conjunction and disjunction operations.*

The most common combinations of disjunction and conjunction are (bounded) sum-product and max-min:

$$\begin{aligned} \mu_{B^*}(y) &= \min\left\{\sum_j^m \left(\mu_{B_j}(y)\cdot\prod_i^{n_j}\mu_{A_{j,i}}(x_i')\right), 1\right\} \\ \mu_{B^*}(y) &= \max_{j\in\{1,\ldots,m\}}\left\{\min\left\{\mu_{A_{j,1}}(x_1'),\ldots,\mu_{A_{j,n_j}}(x_{n_j}'),\mu_{B_j}(y)\right\}\right\} \end{aligned} \tag{A.22}$$

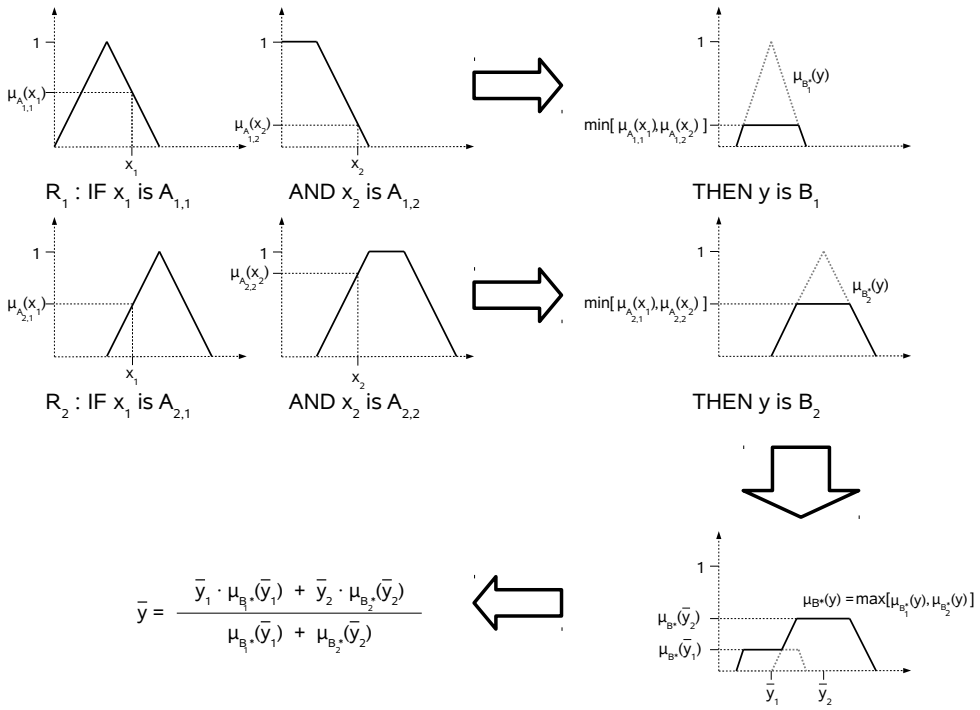


Figure A.2 A schematic representation of a fuzzy logic system. Crisp inputs x_1 and x_2 are fuzzified by the premises of the given rules. For each rule a single conclusion fuzzy set $\mu_{B_1^*}$ and $\mu_{B_2^*}$ is derived by a minimum T-norm. Those conclusions are then combined to the final concluding fuzzy set μ_{B^*} using a max T-conorm. As a last step μ_{B^*} is defuzzified using the height defuzzification. The centers of gravity $\bar{y}_1$ and $\bar{y}_2$ are those from the conclusion fuzzy sets $\mu_{B_1^*}$ and $\mu_{B_2^*}$, i.e. known beforehand. Notice that μ_{B^*} does not need to be computed explicitly when using height defuzzification. Here, it is shown for the sake of completeness.

A fuzzy inference engine maps a (crisp) vector $\vec{x'}$ of premises to a conclusion fuzzy set $\mu_{B^*}(y)$ which assigns a degree of truth to every possible $y \in Y$. Often, one is interested in a single (crisp) representative $\bar{y}$ from the set of possible solutions D_y, which should correspond to some kind of "most typical" solution. The process of deriving such a single, crisp value is called *defuzzification*.

Definition 8 *A defuzzifier is a function $F(D_y) \to D_y$ which maps a fuzzy set $\mu \in F(D_y)$ to a single, crisp value $\bar{y} \in D_y$.*

As for fuzzification, again there exist a wide variety of different *defuzzification* strategies. We will introduce only two of them, the *centroid defuzzifier* and the *height defuzzifier*.

The *centroid defuzzifier* computes the centroid (center of gravity) $\bar{y}$ of the conclusion fuzzy set μ_{B^*}:

$$\bar{y} = \frac{\int_Y y \cdot \mu_{B^*}(y)dy}{\int_Y \mu_{B^*}(y)dy} \tag{A.23}$$

Using $\bar{y}$ as the representative of the conclusion is quite intuitive. It can be seen as the mean or expected conclusion. However, it is quite expensive to compute. The *height defuzzifier* uses the centroids of each individual rule conclusion to determine the defuzzified value of the full result:

$$\bar{y} = \frac{\sum_{j=1}^{m} \bar{y}_j \cdot \mu_{B_j^*}(\bar{y}_j)}{\sum_{j=1}^{m} \mu_{B_j^*}(\bar{y}_j)} \tag{A.24}$$

Notice that when using *height defuzzification* the single rules do not need to be combined to $\mu_{B^*}(y)$ using a T-conorm, as the defuzzified value can be computed directly from the individual rule conclusions. Typically, the center of gravity $\bar{y}_j$ of each single conclusion is known before, so the main computational effort reduces to the evaluation of each $\mu_{B_j^*}(\bar{y}_j)$. From Equation A.19 (or more generally A.18) and the definition of T-norms it is obvious that the center of gravity of any $\mu_{B_j^*}$ equals the center of gravity of μ_{B_j} of the according consequent fuzzy set and thus is not influenced by the parameters $\vec{x'}$.

Definition 9 *A fuzzy logic system (FLS, also called fuzzy logic controller) is a function* $fls : D_1 \times \cdots \times D_n \rightarrow D_y$ *which maps a vector of crisp input values to a crisp output via a fuzzy inference engine (including a fuzzifier) and a defuzzifier.*

The fuzzy inference engine is defined by a set of if-then rules. Each crisp input x_i is discretized by the fuzzy sets used in the premises of the rule set. The evaluation of the inference engine for a given input $\vec{x'}$ yields a fuzzy set $\mu_{B^*}(y)$ which is then defuzzified to a single crisp output $\bar{y}$ (**Figure A.2**).

A fuzzy logic system can be represented as a single formula. Using the product as T-norm and height defuzzification, the according fuzzy logic system is defined as:

$$\bar{y} = fls(x_1, \ldots, x_n) = \frac{\sum_{j=1}^{m} \bar{y}_j \cdot \prod_i^{n_j} \mu_{A_{j,i}}(x_i')}{\sum_{j=1}^{m} \prod_i^{n_j} \mu_{A_{j,i}}(x_i')} \tag{A.25}$$

We have derived the validity of fuzzy reasoning from basic assumptions of propositional logic. The whole process of discretization and reasoning was condensed to a simple, single function - a fuzzy logic system.

Bibliography

[1] Wolkenhauer, O, Kolch, W, and Cho, K. Mathematical systems biology: Genomic cybernetics. In Paton, R, Bolouri, H, Holcombe, M, Parish, J, and Tateson, R, editors, *Computation in cells and tissues. Perspectives an tools of thought*. Springer, 2004.

[2] Ideker, T, Galitski, T, and Hood, L. A new approach to decoding life: Systems biology. *Annual Review of Genomics and Human Genetics*, 2(1):343–372, 2001.

[3] Ilsley, GR, Luscombe, NM, and Apweiler, R. Know your limits: Assumptions, constraints and interpretation in systems biology. *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics*, 1794(9):1280–1287, 2009.

[4] Kremling, A and Saez-Rodriguez, J. Systems biology - An engineering perspective. *Journal of Biotechnology*, 129(2):329–351, 2007.

[5] Westerhoff, HV and Palsson, BO. The evolution of molecular biology into systems biology. *Nature Biotechnology*, 22(10):1249–1252, 2004.

[6] Bruggeman, FJ and Westerhoff, HV. The nature of systems biology. *Trends in Microbiology*, 15(1):45–50, 2007.

[7] Joyce, AR and Palsson, B. The model organism as a system: integrating 'omics' data sets. *Nature Reviews Molecular Cell Biology*, 7(3):198–210, 2006.

[8] Hughes, TR, Mao, M, Jones, AR, Burchard, J, Marton, MJ, Shannon, KW, Lefkowitz, SM, Ziman, M, Schelter, JM, Meyer, MR, Kobayashi, S, Davis, C, Dai, H, He, YD, Stephaniants, SB, Cavet, G, Walker, WL, West, A, Coffey, E, Shoemaker, DD, Stoughton, R, Blanchard, AP, Friend, SH, and Linsley, PS. Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nature Biotechnology*, 19(4):342–347, 2001.

[9] Barrett, J and Kawasaki, ES. Microarrays: the use of oligonucleotides and cDNA for the analysis of gene expression. *Drug Discovery Today*, 8(3):134–141, 2003.

[10] Lamartine, J. The benefits of DNA microarrays in fundamental and applied bio-medicine. *Materials Science and Engineering*, 26(23):354–359, 2006.

[11] Mortazavi, A, Williams, BA, McCue, K, Schaeffer, L, and Wold, B. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature Methods*, 5(7):621–628, 2008.

[12] Cloonan, N, Forrest, ARR, Kolle, G, Gardiner, BBA, Faulkner, GJ, Brown, MK, Taylor, DF, Steptoe, AL, Wani, S, Bethel, G, Robertson, AJ, Perkins, AC, Bruce, SJ, Lee, CC, Ranade, SS, Peckham, HE, Manning, JM, McKernan, KJ, and Grimmond, SM. Stem cell transcriptome profiling via massive-scale mRNA sequencing. *Nature Methods*, 5(7):613–619, 2008.

[13] Morozova, O, Hirst, M, and Marra, MA. Applications of new sequencing technologies for transcriptome analysis. *Annual Review of Genomics and Human Genetics*, 10(1):135–151, 2009.

[14] Reinders, J, Lewandrowski, U, Moebius, J, Wagner, Y, and Sickmann, A. Challenges in mass spectrometry-based proteomics. *Proteomics*, 4(12):3686–3703, 2004.

[15] Xie, F, Liu, T, Qian, WJ, Petyuk, VA, and Smith, RD. Liquid chromatography-mass spectrometry-based quantitative proteomics. *The Journal of Biological Chemistry*, 286(29):25443–25449, 2011.

[16] Griffiths, WJ and Wang, Y. Mass spectrometry: from proteomics to metabolomics and lipidomics. *Chemical Society reviews*, 38(7):1882–1896, 2009.

[17] Bensimon, A, Heck, AJ, and Aebersold, R. Mass spectrometry-based proteomics and network biology. *Annual Review of Biochemistry*, 81(1):379–405, 2012.

[18] Cox, J and Mann, M. Quantitative, high-resolution proteomics for data-driven systems biology. *Annual Review of Biochemistry*, 80:273–299, 2011.

[19] Backer, P, Waele, D, and Speybroeck, L. Ins and outs of systems biology vis-a-vis molecular biology: Continuation or clear cut? *Acta Biotheoretica*, 58(1):15–49, 2009.

[20] Kitano, H. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, 2002.

[21] Bedau, MA. Weak emergence. *Nous*, 31:375–399, 1997.

[22] Anzenbacher, A. *Einführung in die Philosophie*. Herder, 2010.

[23] Haefner, JW. *Modeling Biological Systems: Principles and Applications*. Springer, 2005.

[24] Aldridge, BB, Burke, JM, Lauffenburger, DA, and Sorger, PK. Physicochemical modelling of cell signalling pathways. *Nature Cell Biology*, 8(11):1195–1203, 2006.

[25] Riel, NAWv. Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments. *Briefings in Bioinformatics*, 7(4):364–374, 2006.

[26] Gunawan, R, Gadkar, K, and Doyle III, F. Methods to identify cellular architecture and dynamics from experimental data. In Szallasi, Z, Periwal, V, and Stelling, J, editors, *System Modeling in Cellular Biology. From Concepts to Nuts and Bolts*. MIT Press, 2006.

[27] Lefebvre, C, Rieckhof, G, and Califano, A. Reverse-engineering human regulatory networks. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 4(4):311–325, 2012.

[28] Brazhnik, P, de la Fuente, A, and Mendes, P. Gene networks: how to put the function in genomics. *Trends in Biotechnology*, 20(11):467–472, 2002.

[29] Quo, CF, Kaddi, C, Phan, JH, Zollanvari, A, Xu, M, Wang, MD, and Alterovitz, G. Reverse engineering biomolecular systems using omic data: challenges, progress and opportunities. *Briefings in Bioinformatics*, 13(4):430–445, 2012.

[30] D'haeseleer, P, Liang, S, and Somogyi, R. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[31] Oliver, DJ, Nikolau, B, and Wurtele, ES. Functional genomics: High-throughput mRNA, protein, and metabolite analyses. *Metabolic Engineering*, 4(1):98–106, 2002.

[32] Angel, TE, Aryal, UK, Hengel, SM, Baker, ES, Kelly, RT, Robinson, EW, and Smith, RD. Mass spectrometry-based proteomics: existing capabilities and future directions. *Chemical Society reviews*, 41(10):3912–3928, 2012.

[33] Draghici, S, Khatri, P, Eklund, AC, and Szallasi, Z. Reliability and reproducibility issues in DNA microarray measurements. *Trends in Genetics*, 22(2):101–109, 2006.

[34] Oshlack, A and Wakefield, M. Transcript length bias in RNA-seq data confounds systems biology. *Biology Direct*, 4(1):14, 2009.

[35] Raabe, CA, Hoe, CH, Randau, G, Brosius, J, Tang, TH, and Rozhdestvensky, TS. The rocks and shallows of deep RNA sequencing: Examples in the vibrio cholerae RNome. *RNA*, 17(7):1357–1366, 2011.

[36] Janes, KA and Lauffenburger, DA. A biological approach to computational models of proteomic networks. *Current Opinion in Chemical Biology*, 10(1):73–80, 2006.

[37] Mogilner, A, Wollman, R, and Marshall, WF. Quantitative modeling in cell biology: what is it good for? *Developmental cell*, 11(3):279–287, 2006.

[38] Morelli, LG, Uriu, K, Ares, S, and Oates, AC. Computational approaches to developmental patterning. *Science (New York, NY)*, 336(6078):187–191, 2012.

[39] Wang, W, Cherry, JM, Botstein, D, and Li, H. A systematic approach to reconstructing transcription networks in saccharomyces cerevisiae. *Proceedings of the National Academy of Sciences*, 99(26):16893–16898, 2002.

[40] Xu, X, Wang, L, and Ding, D. Learning module networks from genome-wide location and expression data. *FEBS Letters*, 578(3):297–304, 2004.

[41] Segal, E, Raveh-Sadka, T, Schroeder, M, Unnerstall, U, and Gaul, U. Predicting expression patterns from regulatory sequence in drosophila segmentation. *Nature*, 451(7178):535–540, 2008.

[42] Roth, FP, Hughes, JD, Estep, PW, and Church, GM. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology*, 16(10):939–945, 1998.

[43] Brzma, A, Jonassen, I, Vilo, J, and Ukkonen, E. Predicting gene regulatory elements in silico on a genomic scale. *Genome Research*, 8(11):1202–1215, 1998.

[44] Walker, MG, Volkmuth, W, Sprinzak, E, Hodgson, D, and Klingler, T. Prediction of gene function by genome-scale expression analysis: Prostate cancer-associated genes. *Genome Research*, 9(12):1198–1203, 1999.

[45] Kostka, D and Spang, R. Finding disease specific alterations in the co-expression of genes. *Bioinformatics*, 20(Suppl 1):i194–i199, 2004.

[46] Wolfe, C, Kohane, I, and Butte, A. Systematic survey reveals general applicability of guilt-by-association within gene coexpression networks. *BMC Bioinformatics*, 6(1):227, 2005.

[47] Butte, AJ and Kohane, IS. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing*, pages 418–429, 2000.

[48] Margolin, A, Nemenman, I, Basso, K, Wiggins, C, Stolovitzky, G, Favera, R, and Califano, A. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1):S7, 2006.

[49] Meyer, PE, Kontos, K, Lafitte, F, and Bontempi, G. Information-theoretic inference of large transcriptional regulatory networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007(1):79879, 2007.

[50] Faith, JJ, Hayete, B, Thaden, JT, Mogno, I, Wierzbowski, J, Cottarel, G, Kasif, S, Collins, JJ, and Gardner, TS. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol*, 5(1):e8, 2007.

[51] Küffner, R, Petri, T, Tavakkolkhah, P, Windhager, L, and Zimmer, R. Inferring gene regulatory networks by ANOVA. *Bioinformatics*, 28(10):1376–1382, 2012.

[52] Markowetz, F, Kostka, D, Troyanskaya, OG, and Spang, R. Nested effects models for high-dimensional phenotyping screens. *Bioinformatics*, 23(13):i305–i312, 2007.

[53] Fröhlich, H, Praveen, P, and Tresch, A. Fast and efficient dynamic nested effects models. *Bioinformatics*, 27(2):238–244, 2011.

[54] Tamada, Y, Kim, S, Bannai, H, Imoto, S, Tashiro, K, Kuhara, S, and Miyano, S. Estimating gene networks from gene expression data by combining bayesian network model with promoter element detection. *Bioinformatics*, 19(Suppl 2):ii227–ii236, 2003.

[55] Helman, P, Veroff, R, Atlas, SR, and Willman, C. A bayesian network classification methodology for gene expression data. *Journal of Computational Biology*, 11(4):581–615, 2004.

[56] Pena, JM, Bjorkegren, J, and Tegner, J. Growing bayesian network models of gene networks from seed genes. *Bioinformatics*, 21(Suppl 2):ii224–ii229, 2005.

[57] Tomlin, CJ and Axelrod, JD. Biology by numbers: mathematical modelling in developmental biology. *Nature reviews Genetics*, 8(5):331–340, 2007.

[58] Smolen, P, Baxter, DA, and Byrne, JH. Mathematical modeling of gene networks. *Neuron*, 26(3):567–580, 2000.

[59] de Jong, H. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[60] van Someren, EP, Wessels, LFA, Backer, E, and Reinders, MJT. Genetic network modeling. *Pharmacogenomics*, 3(4):507–525, 2002.

[61] Mandel, J, Palfreyman, NM, Lopez, JA, and Dubitzky, W. Representing bioinformatics causality. *Briefings in Bioinformatics*, 5(3):270–283, 2004.

[62] Schlitt, T and Brazma, A. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(0):1–22, 2007.

[63] Karlebach, G and Shamir, R. Modelling and analysis of gene regulatory networks. *Nat Rev Mol Cell Biol*, 9(10):770–780, 2008.

[64] El Snoussi, H and Thomas, R. Logical identification of all steady states: The concept of feedback loop characteristic states. *Bulletin of Mathematical Biology*, 55(5):973–991, 1993.

[65] Thomas, R, Thieffry, D, and Kaufman, M. Dynamical behaviour of biological regulatory networks - I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bulletin of Mathematical Biology*, 57(2):247–276, 1995.

[66] Pal, R, Ivanov, I, Datta, A, Bittner, ML, and Dougherty, ER. Generating boolean networks with a prescribed attractor structure. *Bioinformatics*, 21(21):4021–4025, 2005.

[67] Klamt, S, Saez-Rodriguez, J, Lindquist, J, Simeoni, L, and Gilles, E. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(1):56, 2006.

[68] Willadsen, K and Wiles, J. Robustness and state-space structure of boolean gene regulatory models. *Journal of Theoretical Biology*, 249(4):749–765, 2007.

[69] Bornholdt, S. Boolean network models of cellular regulation: prospects and limitations. *Journal of the Royal Society, Interface / the Royal Society*, 5 Suppl 1:S85–94, 2008.

[70] Garg, A, Mohanram, K, Di Cara, A, De Micheli, G, and Xenarios, I. Modeling stochasticity and robustness in gene regulatory networks. *Bioinformatics*, 25(12):i101–109, 2009.

[71] Simao, E, Remy, E, Thieffry, D, and Chaouiya, C. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in E.Coli. *Bioinformatics*, 21(Suppl 2):ii190–ii196, 2005.

[72] Sackmann, A, Heiner, M, and Koch, I. Application of Petri net based analysis techniques to signal transduction pathways. *BMC Bioinformatics*, 7:482, 2006.

[73] Lee, DY, Zimmer, R, Lee, SY, and Park, S. Colored petri net modeling and simulation of signal transduction pathways. *Metabolic Engineering*, 8(2):112–122, 2006.

[74] Marwan, W, Wagler, A, and Weismantel, R. A mathematical approach to solve the network reconstruction problem. *Mathematical Methods of Operations Research*, 67(1):117–132, 2008.

[75] Novak, B, Pataki, Z, Ciliberto, A, and Tyson, JJ. Mathematical model of the cell division cycle of fission yeast. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11(1):277–286, 2001.

[76] Kofahl, B and Klipp, E. Modelling the dynamics of the yeast pheromone pathway. *Yeast*, 21(10):831850, 2004.

[77] Sible, JC and Tyson, JJ. Mathematical modeling as a tool for investigating cell cycle control networks. *Methods*, 41(2):238–247, 2007.

[78] Breitling, R, Gilbert, D, Heiner, M, and Orton, R. A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. *Briefings in Bioinformatics*, 9(5):404–421, 2008.

[79] Liebermeister, W, Uhlendorf, J, and Klipp, E. Modular rate laws for enzymatic reactions: thermodynamics, elasticities, and implementation. *Bioinformatics*, 26(12):1528–1534, 2010.

[80] Heiner, M, Koch, I, and Will, J. Model validation of biological pathways using Petri nets - demonstrated for apoptosis. *Biosystems*, 75(1-3):15–28, 2004.

[81] Chen, L, Qi-Wei, G, Nakata, M, Matsuno, H, and Miyano, S. Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets. *Journal of Biosciences*, 32(1):113–127, 2007.

[82] Philippi, N, Walter, D, Schlatter, R, Ferreira, K, Ederer, M, Sawodny, O, Timmer, J, Borner, C, and Dandekar, T. Modeling system states in liver cells: Survival, apoptosis and their modifications in response to viral infection. *BMC Systems Biology*, 3(1):97, 2009.

[83] Li, F, Long, T, Lu, Y, Ouyang, Q, and Tang, C. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.

[84] Chen, KC, Wang, TY, Tseng, HH, Huang, CYF, and Kao, CY. A stochastic differential equation model for quantifying transcriptional regulatory network in saccharomyces cerevisiae. *Bioinformatics*, 21(12):2883–2890, 2005.

[85] Davidich, MI and Bornholdt, S. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3(2), 2008.

[86] Novershtern, N, Regev, A, and Friedman, N. Physical module networks: an integrative approach for reconstructing transcription regulation. *Bioinformatics*, 27(13):i177 –i185, 2011.

[87] Novak, B and Tyson, JJ. A model for restriction point control of the mammalian cell cycle. *Journal of Theoretical Biology*, 230(4):563–579, 2004.

[88] Faure, A, Naldi, A, Chaouiya, C, and Thieffry, D. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–131, 2006.

[89] Gerard, C and Goldbeter, A. Temporal self-organization of the cyclin/Cdk network driving the mammalian cell cycle. *Proceedings of the National Academy of Sciences*, 106(51):21643–21648, 2009.

[90] Sanchez, L and Thieffry, D. A logical analysis of the drosophila gap-gene system. *Journal of Theoretical Biology*, 211(2):115–141, 2001.

[91] Albert, R and Othmer, HG. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *Journal of Theoretical Biology*, 223(1):1–18, 2003.

[92] Tiedemann, HB, Schneltzer, E, Zeiser, S, Rubio-Aliaga, I, Wurst, W, Beckers, J, Przemeck, GK, and Hrabe de Angelis, M. Cell-based simulation of dynamic expression patterns in the presomitic mesoderm. *Journal of Theoretical Biology*, 248(1):120–129, 2007.

[93] Wittmann, DM, Blöchl, F, Trümbach, D, Wurst, W, Prakash, N, and Theis, FJ. Spatial analysis of expression patterns predicts genetic interactions at the mid-hindbrain boundary. *PLoS Comput Biol*, 5(11):e1000569, 2009.

[94] Mendoza, L, Thieffry, D, and Alvarez-Buylla, ER. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15(7-8):593–606, 1999.

[95] Espinosa-Soto, C, Padilla-Longoria, P, and Alvarez-Buylla, ER. A gene regulatory network model for cell-fate determination during arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *The Plant Cell*, 16(11):2923–2939, 2004.

[96] Fisher, J, Piterman, N, Hubbard, EJA, Stern, MJ, and Harel, D. Computational insights into caenorhabditis elegans vulval development. *Proceedings of the National Academy of Sciences*, 102(6):1951–1956, 2005.

[97] Giurumescu, CA, Sternberg, PW, and Asthagiri, AR. Intercellular coupling amplifies fate segregation during caenorhabditis elegans vulval development. *Proceedings of the National Academy of Sciences*, 103(5):1331–1336, 2006.

[98] Bonzanni, N, Krepska, E, Feenstra, KA, Fokkink, W, Kielmann, T, Bal, H, and Heringa, J. Executing multicellular differentiation: quantitative predictive modelling of c.elegans vulval development. *Bioinformatics*, 25(16):2049–2056, 2009.

[99] Teusink, B, Passarge, J, Reijenga, CA, Esgalhado, E, van der Weijden, CC, Schepper, M, Walsh, MC, Bakker, BM, van Dam, K, Westerhoff, HV, and Snoep, JL. Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? Testing biochemistry. *European Journal of Biochemistry*, 267(17):5313–5329, 2000.

[100] Wang, L and Hatzimanikatis, V. Metabolic engineering under uncertainty. I: Framework development. *Metabolic Engineering*, 8(2):133–141, 2006.

[101] Tran, LM, Rizk, ML, and Liao, JC. Ensemble modeling of metabolic networks. *Biophysical Journal*, 95(12):5606–5617, 2008.

[102] Windhager, L and Zimmer, R. Intuitive modeling of dynamic systems with Petri nets and fuzzy logic. In *Lecture Notes in Informatics*, volume P-136, pages 106–115. Gesellschaft für Informatik, 2008.

[103] Windhager, L, Erhard, F, and Zimmer, R. Fuzzy modeling. In Koch, I, Reisig, W, and Schreiber, F, editors, *Modeling in Systems Biology: The Petri Net Approach*, pages 179–208. Springer, 2010.

[104] Stögbauer, T, Windhager, L, Zimmer, R, and Rädler, JO. Experiment and mathematical modeling of gene expression dynamics in a cell-free system. *Integrative Biology*, 4(5):494–501, 2012.

[105] Küffner, R, Petri, T, Windhager, L, and Zimmer, R. Petri nets with fuzzy logic (PNFL): reverse engineering and parametrization. *PLoS ONE*, 5(9):e12807, 2010.

[106] Zadeh, LA. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[107] Mendel, JM. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377, 1995.

[108] Matsuno, H, Tanaka, Y, Aoshima, H, Doi, A, Matsui, M, and Miyano, S. Biopathways representation and simulation on hybrid functional Petri net. *In Silico Biology*, 3(3):389–404, 2003.

[109] Gillespie, DT. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[110] Mangan, S and Alon, U. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.

[111] Novak, B and Tyson, JJ. Design principles of biochemical oscillators. *Nature Reviews Molecular Cell Biology*, 9(12):981–991, 2008.

[112] Tyson, JJ, Chen, KC, and Novak, B. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current Opinion in Cell Biology*, 15(2):221–231, 2003.

[113] Alon, U. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, 2007.

[114] PURExpress kit. New England Biolabs, Germany. http://www.neb.com/nebecomm/products/productE6800.asp, 2010.

[115] Myers, CR, Gutenkunst, RN, and Sethna, JP. Python unleashed on systems biology. *Computing in Science and Engineering*, 9(3):34–37, 2007.

[116] Gutenkunst, RN, Atlas, JC, Casey, FP, Kuczenski, RS, Waterfall, JJ, Myers, CR, and Sethna, JP. SloppyCell. http://sloppycell.sourceforge.net/, 2007.

[117] R Development Core Team. R: A language and environment for statistical computing. http://www.R-project.org, 2011.

[118] Marbach, D, Prill, RJ, Schaffter, T, Mattiussi, C, Floreano, D, and Stolovitzky, G. Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences*, 107(14):6286–6291, 2010.

[119] Marbach, D, Schaffter, T, Mattiussi, C, and Floreano, D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009.

[120] Adler, D. Genetic algorithms and simulated annealing: a marriage proposal. In *, IEEE International Conference on Neural Networks, 1993*, pages 1104–1109 vol.2. IEEE, 1993.

[121] Smith, JE and Fogarty, TC. Operator and parameter adaptation in genetic algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 1(2):81–87, 1997.

[122] D4c2 - dream initiative. http://wiki.c2b2.columbia.edu/dream/index.php/D4c2, 2009.

[123] Christley, S, Nie, Q, and Xie, X. Incorporating existing network information into gene network inference. *PLoS ONE*, 4(8):e6799, 2009.

[124] Mukherjee, S and Speed, TP. Network inference using informative priors. *Proceedings of the National Academy of Sciences*, 105(38):14313 –14318, 2008.

[125] Thomas, R, Paredes, C, Mehrotra, S, Hatzimanikatis, V, and Papoutsakis, E. A model-based optimization framework for the inference of regulatory interactions using time-course DNA microarray expression data. *BMC Bioinformatics*, 8(1):228, 2007.

[126] Deng, X, Geng, H, and Ali, H. EXAMINE: a computational approach to reconstructing gene regulatory networks. *Biosystems*, 81(2):125–136, 2005.

[127] Dietterich, T. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[128] Marbach, D, Mattiussi, C, and Floreano, D. Combining multiple results of a reverse-engineering algorithm: Application to the DREAM five-gene network challenge. *Annals of the New York Academy of Sciences*, 1158(1):102–113, 2009.

[129] Altay, G and Emmert-Streib, F. Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics*, 26(14):1738–1744, 2010.

[130] Agrawal, R, Imieliski, T, and Swami, A. Mining association rules between sets of items in large databases. *SIGMOD Rec*, 22(2):207216, 1993.

[131] The ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, 2012.

[132] Lee, CC. Fuzzy logic in control systems: fuzzy logic controller - part I. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418, 1990.

[133] Michels, K, Klawonn, F, Kruse, R, and Nürnberger, A. *Fuzzy-Regelung. Grundlagen, Entwurf, Analyse*. Springer, 2003.

Acknowledgment

I'm especially grateful to Ralf Zimmer for supervision, funding, and discussions, Robert Küffner for developing the genetic algorithm and successfully applying PNFL during the DREAM4 challenge, Florian Erhard for implementing the PNMA tool, Jonas Zierer for his supporting work as student assistant, Franziska Schneider for administrative support, Gergely Csaba and Frank Steiner for technical support, my dear colleagues for refreshing distractions, and Almut Graebsch for critical assessment of the manuscript, and her love.

Publications

Windhager L, Zimmer R. Intuitive Modeling of Dynamic Systems with Petri Nets and Fuzzy Logic. In *Lecture Notes in Informatics*, volume P-136, pages 106-115. Gesellschaft für Informatik, 2008.

Birzele F, Csaba G, Erhard F, Friedel CC, Küffner R, Petri T, **Windhager L**, Zimmer R. Algorithmische Systembiologie mit Petrinetzen - Von qualitativen zu quantitativen Systemmodellen. *Informatik-Spektrum*, 32(4):310-319, 2009.

Windhager L, Erhard F, Zimmer R. Fuzzy modeling. In Koch I, Reisig W, Schreiber F, editors. *Modeling in Systems Biology: The Petri Net Approach*, pages 179-208. Springer, 2010.

Küffner R, Petri T, **Windhager L**, Zimmer R. Petri Nets with Fuzzy Logic (PNFL): Reverse Engineering and Parametrization. *PLoS One*, 5:e12807, 2010.

Fischer W, **Windhager L**, Rohrer S, Zeiler M, Karnholz A, Hoffmann R, Zimmer R, Haas R. Strain-specific genes of Helicobacter pylori: Genome evolution driven by a novel type IV secretion system and genomic island transfer. *Nucleic Acids Research*, 38(18):6089-6101, 2010.

Marcinowski L, Lidschreiber M, **Windhager L**, Rieder M, Bosse JB, Rädle B, Bonfert T, György I, de Graaf M, da Costa OP, Rosenstiel P, Friedel CC, Zimmer R, Ruzsics Z, Dölken L. Real-time Transcriptional Profiling of Cellular and Viral Gene Expression during Lytic Cytomegalovirus Infection. *PLoS Pathog*, 8(9):e1002908, 2012.

Stögbauer T, **Windhager L**, Zimmer R, Rädler JO. Experiment and Mathematical Modeling of Gene Expression Dynamics in a Cell-Free System. *Integrative Biology*, 4(5):494-501, 2012.

Windhager L, Bonfert T, Burger K, Ruzsics Z, Krebs S, Kaufmann S, Malterer G, LHernault A, Schilhabel M, Schreiber S, Rosenstiel P, Zimmer R, Eick D, Friedel CC, Dölken L. Ultra short and progressive 4sU-tagging reveals key characteristics of RNA processing at nucleotide resolution. *Genome Research*, 22:2031-2042, 2012.

Küffner R, Petri T, Tavakkolkhah P, **Windhager L**, Zimmer R. Inferring Gene Regulatory Networks by ANOVA. *Bioinformatics*, 28(10):1376-1382, 2012.